# THE ELECTRONIC IDENTITY CARD (EID)

## DEVELOPERS GUIDE

VERSION 1.0 EN

***Disclaimer***

# Table Of Contents

# General

This document is intended as a guide for developers to build applications using the eID MW. Using this document, it should be possible to create a simple application that can access the eID card.

Not all methods and objects are described in detail though. Detailed information is provided in the API documentation.

# Prerequisites

To understand the examples given in this document, knowledge of C++, Java and/or C# and some specificities of the languages is required.

# Introduction

The eID MW SDK is a set of libraries used to access both the different kinds of Belgian eID cards and the Belgian social security card (SIS card).
The SDK is built around a C++ core library and is provided for 3 types of operating systems:

- Win32
- Linux
- Mac OSX

To facilitate development of applications in other languages than C++, interfaces have been developed for Java and dotNet.
Most examples in this document will address the Belgian eID card.

## Abbreviations and acronyms

| | | |
|---|---|---|
| eID card | Electronic IDentification card | Any Belgian electronic ID card |
| SIS card | Sociaal Informatie Systeem/ Système Information Sociale | Social security card |
| CRL | Certificate Revocation List | List containing the revoked certificates |
| OCSP | Online Certificate Status Protocol | Protocol used for obtaining the revocation status of a digital certificate |
| RRN | RijksRegister/Registre National | National register |
| UTF-8 | Unicode Transformation Format 8bit | Variable-length character encoding for Unicode |
| QuickInstaller | | |
| APDU | Application Protocol Data Unit | Basic command set for smart cards |
| | | |

## Supported cards

The eID MW SDK supports 2 groups of cards:

- the Belgian eID cards
- Belgian social security card (SIS card).

### Belgian eID Cards

The following eID cards can be read by the SDK:

- Belgian eID card (eID card for Belgian citizens)
- Foreigner eID card (eID card for non Belgian citizens, resident in Belgium)
- Kids Card (eID card for children up to 12 years old)

### Belgian SIS card

The SIS  card is the social security card of each Belgian resident (Belgian or foreigner)

# Compatibility

Platforms:

- Windows: Win32 (Windows 2000, XP, Vista)
- Linux: Fedora 9, Debian etch, OpenSUSE 11
- Mac: OSX 10.4 and 10.5 for PPC and Intel

Programming languages

- C++: Windows/Linux/Mac
- Java: Windows/Linux/Mac
- dotNet languages (VB, C#,…): Windows

C++ compiler:

- Windows: Microsoft Visual Studio 2005
- Linux: default installed g++ compiler
- Mac OSX: default installed g++ compiler

Java:

- JDK 1.4.2 or higher (some samples require 1.5 or higher)

# Installation

In order to use the SDK, the BEID runtime has to be installed first. Download and install the BEID runtime for the required platform and proceed with the installation of the SDK.
To read SIS cards, a special plugin is required. This plugin is installed by the QuickInstaller on Windows and Mac and is always installed on Linux.

## Windows

To install the SDK on Windows, follow the next instructions:

- Download the SDK installation file: BeidMW35-Sdk-<buildnr>.msi

- run the installation by double clicking the msi file and follow the instructions

The SDK is installed in the following directory:

C:\Documents and Settings\<user>\My Documents\Belgium Identity Card SDK

The SIS card plugin must be installed in c:\Windows\system32\ siscardplugins

## Linux

To install the SDK on Linux, follow the next instructions:

- Create a directory where the SDK will be extracted
- Download the appropriate SDK (fedora,OpenSuse,Debian) to this directory
- extract the SDK: tar –xvf beid-sdk-3.5.0-<distro>-<version>-<machinetype>-<buildnr>.tgz

e.g: tar –xvf beid-sdk-3.5.0-fedora-9-i686-1234.tgz

The SIS card plugin must be installed in /usr/local/lib/siscardplugins

## Mac

To install the SDK on Mac, follow the next instructions:

- Create a directory where the SDK must be installed
- Double click the dmg file
- Select the files and folders in the dmg file
- Copy the selected files and directory to the created directory

The SIS card plugin must be installed in /usr/local/lib/siscardplugins

# SDK Description

## General

The SDK can be used in several programming languages as there are: C++,C#, VB, Java. For each language or language group, all necessary interface files and wrappers are delivered.

The BEID 3.5 SDK contains all necessary classes to read the data on the Belgian eID card.
In C++, all methods of the different objects return data in primitives (char, long, etc…) to guarantee the 'one definition' rule in C++ and facilitates the interfacing with other programming languages like Java, C# or VB.

## Components

| C++ | | | Win32 | Linux | Mac |
|---|---|---|---|---|---|
| | eidlib.h | Main eID include file | x | x | x |
| | eidErrors.h | Error codes | x | x | x |
| | eidlibdefines.h | Enumeration types | x | x | x |
| | eidlibException.h | SDK exceptions | x | x | x |
| | beid35libCpp.lib | SDK library to link | x | | |
| | beid35libCpp.dll | SDK main DLL | x | | |
| Java | | | | | |
| | beid35libJava.jar | eID lib Java interface (JNI) | x | x | x |
| | beid35libJava_Wrapper.dll | eID lib Java wrapper dll | x | | |
| | libbeidlibJava_wrapper.so.x.y.z | eID lib Java wrapper shared object | | x | |
| | libbeidlibJava_wrapper.dylib | eID lib JNI wrapper dynamic library | | | x |
| dotNet | | | | | |
| | beid35libCS.dll | eID lib C# interface dll | x | | |
| | beid35libCS_Wrapper.dll | eID lib C# wrapper dll | x | | |
| | | | | | |

## Initializing/Releasing the SDK

For C++, the most important include file to use is '*eidlib.h*'
The eID library is initialized by calling the method *initSDK().*Though Initialization is not mandatory, releasing the SDK must be done using *releaseSDK()*. It is important that the last call to the eID library is the *releaseSDK()* before the application quits. This call will make sure that all background processes are stopped and the memory is cleaned up.
An application should have only one *initSDK()* and one *releaseSDK()*.

Example C++

```
void main( int argc, char* argv[])
{
          BEID_ReaderSet::initSDK()
          // access the eID card here
          …
          BEID_ReaderSet::releaseSDK()
}
```

Example C#

```
BEID_ReaderSet.initSDK();
// access the eID card here
…
BEID_ReaderSet.releaseSDK();
```

Example Java

```
public static void main(String argv[])
{
          if ( -1 != osName.indexOf("Windows") )
          {
                    System.out.println("[Info]  Windows   system!!");
                    System.loadLibrary("beid35libJava_Wrapper");
          }
          else
          {
                    System.loadLibrary("beidlibJava_Wrapper");
          }
          …
          BEID_ReaderSet.initSDK();
          // access the eID card here
          …
          BEID_ReaderSet.releaseSDK();
}
```

# Accessing an eID card

After initialization of the eID library, the system is ready to be used.
To access an eID card, the following sequence must be followed:

- Get the list of card readers
- Select a card reader
- Check a card is present in the card reader
- Depending on the card type, get the card object
- Get the required data object
- Read the data fields

## The readerlist/readercontext

The reader list class *BEID_ReaderSet* represents the list of card readers connected to the machine and has a variety of functions to obtain information about the connected card readers. From this reader list, a card reader can be selected and the resulting reader context *BEID_ReaderContext* can be used to access the card in the card reader.

Example C++:

```
…
unsigned long          nrReaders  = ReaderSet.readerCount();
const char* const*     readerList = ReaderSet.readerList();
for ( size_t readerIdx=0; readerIdx<nrReaders; readerIdx++)
{
        BEID_ReaderContext& readerContext = ReaderSet.getReaderByName(readerList[readerIdx]);
        bool bCardPresent = readerContext.isCardPresent();
…
}
…
```

Example Java:

```
…
long nrReaders = BEID_ReaderSet.instance().readerCount();
String[] readerList = BEID_ReaderSet.instance().readerList();
for (int readerIdx = 0; readerIdx < nrReaders; readerIdx++)
{
        BEID_ReaderContext readerContext =
                BEID_ReaderSet.instance().getReaderByName(readerList[readerIdx]);
        boolean bCardPresent = readerContext.isCardPresent();
        …
}
…
```

Example C#:

```
…
long nrReaders = BEID_ReaderSet.instance().readerCount();
string[] readerList = BEID_ReaderSet.instance().readerList();
for (int readerIdx = 0; readerIdx < nrReaders; readerIdx++)
{
        BEID_ReaderContext readerContext =
                BEID_ReaderSet.instance().getReaderByName(readerList[readerIdx]);
        bool bCardPresent = readerContext.isCardPresent();
        …
}
…
```

**Note:** A quick way to get a reader context is the method *ReaderSet.getReader()*. This method will get the reader context of the first reader that has a card or, when no cards are inserted, the reader context of the first card reader.

## Obtaining the card object

The reader context allows access to the card (if present). Once the card type is determined via the reader context (*`getCardType()`*), the corresponding card object can be obtained (*`getEIDCard(),getForeignerCard(),getKidsCard(),getSISCard()`*)

**Note:** Since foreigner and kids cards are eID cards, the method *getEIDCard()* can be used for any eID type card.

Example in C++
We assume a reader context is obtained:

```
…
switch(readerContext.getCardType())
{
case BEID_CARDTYPE_EID:
        {
        BEID_EIDCard& card = readerContext.getEIDCard()
        std::string name( card.getID().getSurname());
        …
        }
        break;
case BEID_CARDTYPE_KIDS:
        {
        BEID_KidsCard& card = readerContext.getKidsCard();
        …
        }
        break;
case BEID_CARDTYPE_FOREIGNER:
        {
        BEID_ForeignerCard& card = readerContext.getForeignerCard();
        …
        }
        break;
case BEID_CARDTYPE_SIS:
        {
        BEID_SISCard& card = readerContext.getSISCard();
        …
        }
        break;
case BEID_CARDTYPE_UNKNOWN:
default:
        // unknown card type
        …
        break;
}
…
```

Example Java:

```
…
BEID_CardType  cardType = readerContext.getCardType();

if (cardType == BEID_CardType.BEID_CARDTYPE_EID)
{
        BEID_EIDCard card = readerContext.getEIDCard();
        …
}
else if (cardType == BEID_CardType.BEID_CARDTYPE_KIDS)
{
        BEID_KidsCard card = readerContext.getKidsCard();
        …
}
else if (cardType == BEID_CardType.BEID_CARDTYPE_FOREIGNER)
{
        BEID_ForeignerCard card = readerContext.getForeignerCard();
        …
}
else if (cardType == BEID_CardType.BEID_CARDTYPE_SIS)
{
        BEID_SISCard card = readerContext.getSISCard();
        …
}
else
{
        …
}
…
```

Example C#:

```
…
switch(readerContext.getCardType())
{
case BEID_CardType.BEID_CARDTYPE_EID:
        {
        BEID_EIDCard card = readerContext.getEIDCard()
        std::string name( card.getID().getSurname());
        …
        }
        break;
case BEID_CardType.BEID_CARDTYPE_KIDS:
        {
        BEID_KidsCard card = readerContext.getKidsCard();
        …
        }
        break;
case BEID_CardType.BEID_CARDTYPE_FOREIGNER:
        {
        BEID_ForeignerCard card = readerContext.getForeignerCard();
        …
        }
        break;
case BEID_CardType.BEID_CARDTYPE_SIS:
        {
        BEID_SISCard card = readerContext.getSISCard();
        …
        }
        break;
case BEID_CardType.BEID_CARDTYPE_UNKNOWN:
default:
        // unknown card type
        …
        break;
}
…
```

## Reading the data

The data on a card is stored in multiple ‘documents’:

- ID document: contains the persons data
- Picture document: contains the persons picture
- VersionInfo document: contains version information of the card

**Note:** Text data retrieved from the eID card is in UTF-8.

### *ID data*

The ID data is a collection of information on the person’s identity like

- Name, firstname
- Gender
- Birth data and place
- Address
- …

To obtain the ID data object, the method *getID()* can be used. This object can then be used to access the different id data fields like name, address, etc…

Example in C++:

```
…
BEID_EIDCard& card = readerContext.getEIDCard();
BEID_EId&     eid  = card.getID();
std::cout << "\tFirstName          : " << eid.getFirstName()             << std::endl;
std::cout << "\tSurname            : " << eid.getSurname()               << std::endl;
std::cout << "\tGender             : " << eid.getGender()                << std::endl;
std::cout << "\tDateOfBirth        : " << eid.getDateOfBirth()           << std::endl;
std::cout << "\tLocationOfBirth    : " << eid.getLocationOfBirth()       << std::endl;
std::cout << "\tNationality        : " << eid.getNationality()           << std::endl;
std::cout << "\tNationalNumber     : " << eid.getNationalNumber()        << std::endl;
std::cout << "\tSpecialOrganization: " << eid.getSpecialOrganization()   << std::endl;
std::cout << "\tMemberOfFamily     : " << eid.getMemberOfFamily()        << std::endl;
std::cout << "\tAddressVersion     : " << eid.getAddressVersion()        << std::endl;
std::cout << "\tStreet             : " << eid.getStreet()                << std::endl;
std::cout << "\tZipCode            : " << eid.getZipCode()               << std::endl;
std::cout << "\tMunicipality       : " << eid.getMunicipality()          << std::endl;
std::cout << "\tCountry            : " << eid.getCountry()               << std::endl;
std::cout << "\tSpecialStatus      : " << eid.getSpecialStatus()         << std::endl;
…
```

Example Java:

```
…
BEID_EIDCard card = readerContext.getEIDCard();
BEID_EId     eid  = card.getID();
System.out.println( "\tFirstName          : " + eid.getFirstName());
System.out.println( "\tSurname            : " + eid.getSurname());
System.out.println( "\tGender             : " + eid.getGender());
System.out.println( "\tDateOfBirth        : " + eid.getDateOfBirth());
System.out.println( "\tLocationOfBirth    : " + eid.getLocationOfBirth());
System.out.println( "\tNationality        : " + eid.getNationality());
System.out.println( "\tNationalNumber     : " + eid.getNationalNumber());
System.out.println( "\tSpecialOrganization: " + eid.getSpecialOrganization());
System.out.println( "\tMemberOfFamily     : " + eid.getMemberOfFamily());
System.out.println( "\tAddressVersion     : " + eid.getAddressVersion());
System.out.println( "\tStreet             : " + eid.getStreet());
System.out.println( "\tZipCode            : " + eid.getZipCode());
System.out.println( "\tMunicipality       : " + eid.getMunicipality());
System.out.println( "\tCountry            : " + eid.getCountry());
System.out.println( "\tSpecialStatus      : " + eid.getSpecialStatus());
…
```

Example C#:

```
…
BEID_EIDCard card;
card = Reader.getEIDCard();

BEID_EId doc;
doc = card.getID();

string sText;
sText = "";
sText += "First Name = " + doc.getFirstName() + "\r\n";
sText += "Last Name = " + doc.getSurname() + "\r\n";
sText += "Gender = " + doc.getGender() + "\r\n";
sText += "DateOfBirth = " + doc.getDateOfBirth() + "\r\n";
sText += "LocationOfBirth = " + doc.getLocationOfBirth() + "\r\n";
sText += "Nobility = " + doc.getNobility() + "\r\n";
sText += "Nationality = " + doc.getNationality() + "\r\n";
sText += "NationalNumber = " + doc.getNationalNumber() + "\r\n";
sText += "SpecialOrganization = " + doc.getSpecialOrganization() + "\r\n";
sText += "MemberOfFamily = " + doc.getMemberOfFamily() + "\r\n";
sText += "AddressVersion = " + doc.getAddressVersion() + "\r\n";
sText += "Street = " + doc.getStreet() + "\r\n";
sText += "ZipCode = " + doc.getZipCode() + "\r\n";
sText += "Municipality = " + doc.getMunicipality() + "\r\n";
sText += "Country = " + doc.getCountry() + "\r\n";
sText += "SpecialStatus = " + doc.getSpecialStatus() + "\r\n";
…
```

### *Picture data*

The picture object contains the image data of the owner of the eID card. This picture is a file in jpg format.
To obtain the picture object, the method *getPicture()* can be used.

**Note:** a SIS card has no picture.

Example C++:

```
…
BEID_EIDCard& Card = readerContext.getEIDCard();
BEID_Picture& Picture = Card.getPicture();
const unsigned char* PictureData = Picture.getData().GetBytes();

// now the data can be written to a file
…
```

Example Java:

```
…
BEID_EIDCard Card = readerContext.getEIDCard();
BEID_Picture Picture = Card.getPicture();
byte[] PictureData = Picture.getData().GetBytes();

// now the data can be written to a file
…
```

Example C#:

```
…
BEID_Picture picture;
picture = card.getPicture();

byte[] bytearray;
bytearray = picture.getData().GetBytes();

// now the data can be written to a file
…
```

### *Version Information data*

This document contains a collection of data of the eID card itself, like serial number, etc…

The version document object can be obtained with the method *getVersionInfo()*. The version object has access methods for each field of the version information.

Example in C++:

```
…
BEID_EIDCard& Card = readerContext.getEIDCard();
BEID_CardVersionInfo& CardVersionInfo = Card.getVersionInfo();
…= CardVersionInfo.getSerialNumber();
…= CardVersionInfo.getComponentCode();
…= CardVersionInfo.getOsNumber();
…= CardVersionInfo.getOsVersion();
…= CardVersionInfo.getSoftmaskNumber();
…= CardVersionInfo.getSoftmaskVersion();
…= CardVersionInfo.getAppletVersion();
…= CardVersionInfo.getGlobalOsVersion();
…= CardVersionInfo.getAppletInterfaceVersion();
…= CardVersionInfo.getPKCS1Support();
…= CardVersionInfo.getKeyExchangeVersion();
…= CardVersionInfo.getAppletLifeCycle();
…= CardVersionInfo.getGraphicalPersonalisation();
…= CardVersionInfo.getElectricalPersonalisationInterface();
…
```

Example Java:

```
…
BEID_EIDCard Card = readerContext.getEIDCard();
BEID_CardVersionInfo CardVersionInfo = Card.getVersionInfo();
…= CardVersionInfo.getSerialNumber();
…= CardVersionInfo.getComponentCode();
…= CardVersionInfo.getOsNumber();
…= CardVersionInfo.getOsVersion();
…= CardVersionInfo.getSoftmaskNumber();
…= CardVersionInfo.getSoftmaskVersion();
…= CardVersionInfo.getAppletVersion();
…= CardVersionInfo.getGlobalOsVersion();
…= CardVersionInfo.getAppletInterfaceVersion();
…= CardVersionInfo.getPKCS1Support();
…= CardVersionInfo.getKeyExchangeVersion();
…= CardVersionInfo.getAppletLifeCycle();
…= CardVersionInfo.getGraphicalPersonalisation();
…= CardVersionInfo.getElectricalPersonalisationInterface();
…
```

Example C#:

```
…
BEID_EIDCard Card = readerContext.getEIDCard();
BEID_CardVersionInfo CardVersionInfo = Card.getVersionInfo();
…= CardVersionInfo.getSerialNumber();
…= CardVersionInfo.getComponentCode();
…= CardVersionInfo.getOsNumber();
…= CardVersionInfo.getOsVersion();
…= CardVersionInfo.getSoftmaskNumber();
…= CardVersionInfo.getSoftmaskVersion();
…= CardVersionInfo.getAppletVersion();
…= CardVersionInfo.getGlobalOsVersion();
…= CardVersionInfo.getAppletInterfaceVersion();
…= CardVersionInfo.getPKCS1Support();
…= CardVersionInfo.getKeyExchangeVersion();
…= CardVersionInfo.getAppletLifeCycle();
…= CardVersionInfo.getGraphicalPersonalisation();
…= CardVersionInfo.getElectricalPersonalisationInterface();
…
```

## *eID PIN Codes*

An eID card contains a PIN code. This PIN code can be verified or changed using the *BEID_Pins* object .

The most important methods to access the PIN are:

- *verifyPin()*: check the given PIN code
- *changePin()*: change the PIN code to a new code

**Note:** Each failed attempt to access the PIN code will result in a decrement of the number of tries left. After 3 failed attempts, the Belgian eID card will be blocked. It is not possible to unblock the eID card with the middleware.

Example in C++:

```
…
BEID_ReaderContext &reader = ReaderSet.getReader();
BEID_EIDCard &card = reader.getEIDCard();
if(card.getPins().getPinByNumber(0).verifyPin("",ulRemaining))
{
        // success
}
else
{
        if(ulRemaining==0xFFFF)
        // verify pin canceled
        else
        //verify pin failed
}
```

Example Java:

```
…
BEID_ReaderContext reader = BEID_ReaderSet.instance().getReader();
BEID_EIDCard card = reader.getEIDCard();
BEID_ulwrapper ulRemaining = new BEID_ulwrapper(-1);
if(card.getPins().getPinByNumber(0).verifyPin("",ulRemaining))
{
        // success
}
else
{
        if(ulRemaining.m_long == -1)
        // verify pin canceled
        else
        //verify pin failed
}
…
```

Example C#:

```
…
BEID_ReaderSet ReaderSet;
ReaderSet = BEID_ReaderSet.instance();
BEID_ReaderContext Reader;
Reader = ReaderSet.getReader();
uint lRemaining=0;
if (Reader.getEIDCard().getPins().getPinByNumber(0).verifyPin("",ref lRemaining))
{
        // success
}
else
{
        if(ulRemaining == -1)
        // verify pin canceled
        else
        //verify pin failed
}
…
```

## *Certificates*

The certificates on the eID card can be obtained using the *getCertificates()* method on the eID card object. This method will return a certificate container object *BEID_Certificates*. From this object, all certificates can be accessed and their related data can be retrieved like:

- Serial number
- Owner of the certificate
- Issuer of the certificate
- Validity dates of the certificate
- …

Certificates can be added to the certificate container (*addCertificate()*).

From the certificate container, individual certificates can be selected in different ways:
Index based:

- *getCertFromCard(unsigned long ulIndexCard)*
- *getCert(unsigned long ulIndexAll)*

**Note:** *getCert(…)* allows to access all certificates in the container while *getCertFromCard(…)* will only address the certificates from the eID card. The method *countAll()* should be used with *getCert()* and *countFromCard()* should be used together with *getCertFromCard()*.

Type based:

- *getCert(BEID_CertifType type)*

Direct access:

- *getRrn()*
- *getRoot()*
- *getCA()*
- *getSignature()*
- *getAuthentication()*

As soon as the status of a certificate is requested (‘*BEID_Certificate::getStatus(...)*’), the certificates will be verified using either the Certificate Revocation List (CRL) or the Online Certificate Status Protocol(OCSP). When no internet connection is available, the method will immediately return and set the status accordingly.

**Note:** With the CRL verification set to ‘mandatory’, the eID MW will download several megabytes of data. Depending on the speed of the internet connection, this download can take several seconds.

Example in C++

```
…
BEID_EIDCard&          Card           = ReaderContext.getEIDCard();
BEID_Certificates&     certificates   = Card.getCertificates();

for (size_t CertIdx=0;CertIdx<Card.certificateCount();CertIdx++)
{
       BEID_Certificate&      cert = certificates.getCertFromCard(CertIdx);
       …
}
…
```

Example Java:

```
…
BEID_EIDCard           Card           = readerContext.getEIDCard();
BEID_Certificates      certificates   = Card.getCertificates();

for (size_t CertIdx=0;CertIdx<Card.certificateCount();CertIdx++)
{
       BEID_Certificate cert = certificates.getCertFromCard(CertIdx);
       …
}
…
```

Example C#:

```
…
BEID_EIDCard           Card           = readerContext.getEIDCard();
BEID_Certificates      certificates   = Card.getCertificates();

for (size_t CertIdx=0;CertIdx<Card.certificateCount();CertIdx++)
{
       BEID_Certificate cert = certificates.getCertFromCard(CertIdx);
       …
}
…
```

# Card status events (callbacks)

The eID library has a callback mechanism implemented that allows the library to call the application each time the status of the card in the card reader is changed (card inserted or retracted). The callback mechanism can be setup per card reader connected to the machine.

**Note:** Depending on the programming language used, this mechanism is implemented in a slightly different way.

## Setting up the callback mechanism

In C++, to each card reader a C-style function can be associated by means of a function pointer. With the function, a data pointer can be passed holding user data.
When the status of the card changes (card retracted or inserted) the callback function is called by the eID library with the data pointer as one of the parameters.

Activating the callback mechanism is done with the method *BEID_ReaderContext::SetEventCallback(…)* which returns an event handle. This handle should be stored and used to stop the event callback (see further).

Example C++:

```
// callback function
void myCallback (…)
{
        // we end up here at callback time
}

// user data class
class MyData
{
…
};

size_t maxcount=ReaderSet.readerCount(true);
for (size_t Ix=0; Ix<maxcount; Ix++)
{
void (*fCallback)(…);
        const char*             readerName     = ReaderSet.getReaderName(Ix);
        BEID_ReaderContext&     readerContext  = ReaderSet.getReaderByNum(Ix);
        MyData*                 pData          = new MyData (…);
        fCallback = (void (*)(…))&myCallback;
        unsigned long eventHandle = readerContext.SetEventCallback(fCallback,pData);
        // handle must be kept!!!
        …
}
```

In Java, a callback mechanism with function pointers like in C++ does not exist. In order to provide a similar mechanism, an interface class ‘*Callback*’ is provided by the SDK. This interface contains one method ‘`getEvent(…)`’.
At the application side, a class has to be defined that implements the interface class and as such the method ‘*getEvent(…)*’.

Example Java: implementing the Callback interface:

```
public class EventCallback implements Callback
{
…
       public void getEvent(long lRet, long ulState, Object data)
       {
       // we end up here as callback
       …
       }
}
```

As in the C++ version, a user data object can be passed as well. This data object must be derived from ‘object’ and as such, primitive types are not allowed.

Example Java: defining a user data class:

```
public class EventData
{
       …
       EventData(String readerName)
       {
              m_readerName = readerName;
       }
       public String m_readerName;
       …
}
```

This user defined callback object, together with the user defined data must be passed in the method ‘`BEID_ReaderContext:`*:SetEventCallback(...)*’ to setup the callback mechanism.
When the card status changes, the callback mechanism will call the method ‘*getEvent(...)*’ with the user data as a parameter.

Example Java: Setting the callback:

```
public class main
{
   …
   public static void main(String[] args)
   {
       …
       for (int readerIdx = 0
; readerIdx < BEID_ReaderSet.instance().readerCount()
; readerIdx++
            )
       {
       BEID_ReaderContext readerContext =
              BEID_ReaderSet.instance().getReaderByNum(readerIdx);
       String readerName = readerContext.getName();
       EventCallback eventCallBack = new EventCallback();
       EventData eventData = new EventData(readerName);
       long handle = readerContext.SetEventCallback(eventCallBack, eventData);
       // handle must be kept!!
       …
       }
       …
   }
   …
```

In C#, the callback mechanism is very similar to C++.
First of all, a callback function must be declared. The signature for the callback function is predefined by the delegate ‘*BEID_SetEventDelegate()*’.

Example C#: defining a callback function:

```
// callback function
public void CallBack(int lRe, uint lState, System.IntPtr p)
{
        …
        try
        {
                …
                string readerName;
                ReaderRef readerRef;

                readerName =
                   System.Runtime.InteropServices.Marshal.PtrToStringAnsi(p);
                readerRef = (ReaderRef)MyReadersSet[readerName];

                if(readerRef.reader.isCardPresent())
                 {
                     if(readerRef.reader.isCardChanged(ref readerRef.cardId))
                     {
                       …
                     }
                 }
                 else
                 {
                     if(readerRef.cardId != 0)
                     {
                       …
                     }
                 }
                …
        }
        catch (BEID_Exception ex)
        {
        …
        }
        catch (Exception ex)
        {
        …
        }
}
```

To allow the callback mechanism to use the callback function, it must be assigned to a delegate *`BEID_SetEventDelegate`(…)*. Hooking in the callback function is done with the *BEID_ReaderContext::SetEventCallback()*.
User data can be passed around as unmanaged data. How this can be achieved is shown in the next code snippet.

Example C# assigning the callback to the delegate and setting the callback:

```
private class ReaderRef
{
        public BEID_ReaderContext reader;
        public uint eventHandle;
        public IntPtr ptr;
        public uint cardId;
}
System.Collections.Hashtable MyReadersSet = new System.Collections.Hashtable();
private void AttachEvents()
{
        try
        {
        BEID_ReaderContext reader;
        ReaderRef readerRef;
        uint i;
        BEID_SetEventDelegate MyCallback= new BEID_SetEventDelegate(CallBack);
        string readerName;
        for(i = 0;i<ReaderSet.readerCount();i++)
                {
                reader = ReaderSet.getReaderByNum(i);
                readerName = ReaderSet.getReaderName(i);
                readerRef = new ReaderRef();
                readerRef.reader = reader;
                readerRef.ptr =
                   System.Runtime.InteropServices.Marshal.StringToHGlobalAnsi(readerName);
rRef.cardId = 0;
                MyReadersSet.Add(readerName, readerRef);
                readerRef.eventHandle = reader.SetEventCallback(MyCallback, readerRef.ptr);
                }

        }
        catch (BEID_Exception ex)
        {
                …
        }
        catch (Exception ex)
        {
                …
        }

}
```

## Stopping the callback mechanism

With the method *BEID_ReaderContext ::StopEventCallback()* the callback mechanism can be stopped. This should be done for each card reader (reader context) separately.

The methods parameter is the handle that the *SetEventCallback()* returned. In the next sample it is assumed this handle is available.

Example C++:

```
…
size_t maxcount = ReaderSet.readerCount();
for (size_t Ix=0; Ix<maxcount; Ix++)
{
        BEID_ReaderContext&   readerContext  = ReaderSet.getReaderByNum(Ix);
        //recover the handle
        Unsigned long eventHandle = …;
        readerContext.StopEventCallback (eventHandle);
        …
}
...
```

**Note:** In this example, the *readerCount()* method is used (same as default parameter set to false) . This is done to make sure the number of readers is not updated, just before the callbacks are stopped.

Example in Java:

```
…
long maxcount = BEID_ReaderSet.instance().readerCount();
for (long Ix=0; Ix<maxcount; Ix++)
{
        BEID_ReaderContext    readerContext  = BEID_ReaderSet.instance().getReaderByNum(Ix);
        // recover the handle
        long eventHandle = …;
        readerContext.StopEventCallback (eventHandle);
        …
}
…
```

The same procedure is valid also for C#.

Example in C#:

```
…
// use the container to recover the event handle
foreach(ReaderRef readerRef in MyReadersSet.Values)
{
        reader = readerRef.reader;
        reader.StopEventCallback(readerRef.eventHandle);
}
…
```

## Waiting for events

This callback mechanism can be used to create blocking calls that wait until a card is present or a card is removed from the card reader (see also the sample *wait_card*).

To implement this, two methods could be defined, *WaitForCardAbsent()* and *WaitForCardPresent()* in analogy to the methods in the Java *CardTerminal* object:

Example C++: blocking calls:

```
class TimeoutException : public std::exception
{
public:
       TimeoutException()
       {
       }
       virtual const char* what() const throw()
       {
           …
       }
};



BEID_ReaderContext& WaitForCardPresent(int Timeout)
{
    BEID_ReaderContext *reader=NULL;
    try
    {
        bool bContinue  = true;
        int  Count           = 0;
        while(bContinue)
        {
            reader = &m_ReaderSet->getReader();
            if(reader->isCardPresent())
            {
                bContinue=false;
            }
            else if (Count > Timeout)
            {
               bContinue=false;
               TimeoutException *e = new TimeoutException();
               throw(e);
            }
            else
            {
#ifdef WIN32
               Sleep(1000);
#else
               usleep(1000 * 1000);
#endif
                Count++;
            }
        }
       if(!bContinue)
               return *reader;
    }
    catch (BEID_ExNoReader &ex)
    {   …    }
    catch (BEID_Exception &ex)
    {   …    }
    catch ( TimeoutException* ex)
    {   throw;    }
    catch (...)
    {
        …
        throw;
    }

    throw(new std::exception());
}
```

```
bool WaitForCardAbsent(int Timeout)
{
    try
    {
        BEID_ReaderContext* reader = NULL;
        Int Count  = 0;
        while (true)
        {
            reader = &m_ReaderSet->getReader();

            if (!reader->isCardPresent())
            {
                return true;
            }
            else if (Count > Timeout)
            {
                return false;
            }
            else
            {
#ifdef WIN32
               Sleep(1000);
#else
               usleep(1000 * 1000);
#endif
               Count++;
            }
        }
    }
    catch (BEID_ExNoReader &ex)
    {
        …
    }
    catch (BEID_Exception &ex)
    {
        …
    }
    catch (...)
    {
        …
    }

    return false;
}
```

These 2 functions can then be used as blocking calls in an application as follows:

Example using the blocking calls C++:

```
…
try
{
        BEID_ReaderContext &reader = WaitForCardPresent(_TIMEOUT);
        BEID_EIDCard &card = reader.getEIDCard();
        …
        WaitForCardAbsent(_TIMEOUT);
        …
}
catch(TimeoutException *ex)
{
        …
}
…
```

In Java, the same behavior can be achieved.

Example Java: blocking calls:

```
private static BEID_ReaderContext WaitForCardPresent(int Timeout) throws
BEID_ExNoReader,BEID_Exception,Exception
{
   int   Count   = 0;
   boolean bContinue = true;
   boolean bRetVal      = false;
   BEID_ReaderContext reader = null;
   while(bContinue)
   {
      reader = BEID_ReaderSet.instance().getReader();
      if(reader.isCardPresent())
      {
         bRetVal = true;
         break;
      }
      else if (Count > Timeout)
      {
         bRetVal = false;
         break;
      }
      else
      {
         Thread.currentThread().sleep(1000);
         Count++;
      }
   }
   if( bRetVal )
   {
      return reader;
   }
   Exception e = new Exception("Timeout");
   throw(e);
}
private static boolean WaitForCardAbsent(int Timeout) throws
BEID_ExNoReader,BEID_Exception,Exception
{
   BEID_ReaderContext reader = null;
   int Count = 0;
   while (true)
   {
      reader = BEID_ReaderSet.instance().getReader();
      if (!reader.isCardPresent())
      {
         return true;
      }
      else if (Count > Timeout)
      {
         return false;
      }
      else
      {
         Thread.currentThread().sleep(1000);
         Count++;
      }
   }
}
```

In C#, the same idea stands in the following example.

Example C#: blocking calls:

```
private BEID_ReaderContext WaitForCardPresent(int Timeout)
{
        BEID_ReaderContext reader=null;
        try
        {
                bool bContinue = true;
                int Count=0;

                while(bContinue)
                {
                        reader = m_ReaderSet.getReader();
                        if(reader.isCardPresent())
                        {
                                bContinue=false;
                        }
                        else if (Count > Timeout)
                        {
                                bContinue = false;
                                reader = null;
                        }
                        else
                        {
                                Thread.Sleep(1000);
                                Count++;
                        }
                }
                return reader;
        }
        catch (BEID_ExNoReader ex)
        {…}
        catch (BEID_Exception ex)
        {…}
        catch (Exception ex)
        {…}
        return null;
}
private bool WaitForCardAbsent(int Timeout)
{
        try
        {
                BEID_ReaderContext reader = null;
                int Count = 0;
                while (true)
                {
                        reader = m_ReaderSet.getReader();
                        if (!reader.isCardPresent())
                        {
                                return true;
                        }
                        else if (Count > Timeout)
                        {
                                return false;
                        }
                        else
                        {
                                Thread.Sleep(1000);
                                Count++;
                        }
                }
        }
        catch (BEID_ExNoReader ex)
        {…}
        catch (BEID_Exception ex)
        {…}
        catch (Exception ex)
        {…}
        return false;
}
```

# Certificate validation

As mentioned before, certificates are stored on the eID card. These certificates have a validity which can be verified in 3 ways:

- Not validated: the signature of the certificate is verified and the root certificate must be a valid Belgian root.
- Via CRL : The CRL is downloaded and the MW checks if the certificate is in the revocation list or not, meaning revoked or valid.
- Via OCSP : A request is sent to the OCSP responder which answers weather the certificate is valid, revoked or unknown.

OCSP validation needs a permanent internet connection, while CRL validation needs a connection when a new CRL is requested.

Validation of a certificate may take several seconds or even several minutes. So to avoid that all applications based on the middleware hang each time they request a certificate status, the MW shares the statuses through a caching mechanism. Each status remains valid for 15 minutes, and then it’s renewed if required again.

For the same reason, download of the CRL is managed only by one process and shared by the MW with other applications. (A CRL is updated several time per day on the internet repository but its validity is about one week. So if a previous downloaded CRL is still valid, it won’t be downloaded again until the end of  its validity).

The status of a certificate can be obtained by different methods:

- *`getStatus()`*: returns the certificate status using the user validation parameters (from the registry/config file).
- *`getStatus(BEID_ValidationLevel crl, BEID_ValidationLevel ocsp)`*: returns the certificate status according the specified validation parameter.
- *`verifyCRL()`*: returns the status using the CRL validation.
- *`verifyOCSP()`*: returns the status using the OCSP validation.

Example C++

```
BEID_ReaderContext &reader=ReaderSet.getReader();
BEID_EIDCard &card=reader.getEIDCard();
BEID_Certificates &store=card.getCertificates();
BEID_Certificate &certAuth=store.getAuthentication();
BEID_CertifStatus status=certAuth.getStatus();

if(status==BEID_CERTIF_STATUS_VALID)
          std::cout << "The authentication certificate is issued from a trusted Belgian root" << std::endl;
else if(status==BEID_CERTIF_STATUS_VALID_CRL)
          std::cout << "The authentication certificate has been succesfully verify through CRL validation" << std::endl;
else if(status==BEID_CERTIF_STATUS_VALID_OCSP)
          std::cout << "The authentication certificate has been succesfully verify through OCSP validation" << std::endl;
else if(status==BEID_CERTIF_STATUS_REVOKED)
          std::cout << "The authentication certificate is revoked" << std::endl;
else
          std::cout << "The authentication certificate could not be validated" << std::endl;
```

Example C#

```
BEID_ReaderSet ReaderSet = BEID_ReaderSet.instance();
BEID_ReaderContext Reader = ReaderSet.getReader();
BEID_EIDCard card = Reader.getEIDCard();
BEID_Certificates store = card.getCertificates();
BEID_Certificate certAuth=store.getAuthentication();
BEID_CertifStatus status=certAuth.getStatus();
string textStatus;

if (status == BEID_CertifStatus.BEID_CERTIF_STATUS_VALID)
          textStatus = "The authentication certificate is issued from a trusted Belgian root";
else if (status == BEID_CertifStatus.BEID_CERTIF_STATUS_VALID_CRL)
          textStatus = "The authentication certificate has been succesfully verify through CRL validation";
else if (status == BEID_CertifStatus.BEID_CERTIF_STATUS_VALID_OCSP)
          textStatus = "The authentication certificate has been succesfully verify through OCSP validation";
else if (status == BEID_CertifStatus.BEID_CERTIF_STATUS_REVOKED)
          textStatus = "The authentication certificate is revoked";
else
          textStatus = "The authentication certificate could not be validated";
```

Example Java

```
BEID_ReaderSet ReaderSet = BEID_ReaderSet.instance();
BEID_ReaderContext Reader = ReaderSet.getReader();
BEID_EIDCard card = Reader.getEIDCard();
BEID_Certificates store = card.getCertificates();
BEID_Certificate certAuth = store.getAuthentication();
BEID_CertifStatus status = certAuth.getStatus();
String textStatus;

if (status == BEID_CertifStatus.BEID_CERTIF_STATUS_VALID)
          textStatus = "The authentication certificate is issued from a trusted Belgian root";
else if (status == BEID_CertifStatus.BEID_CERTIF_STATUS_VALID_CRL)
          textStatus = "The authentication certificate has been succesfully verify through CRL validation";
else if (status == BEID_CertifStatus.BEID_CERTIF_STATUS_VALID_OCSP)
          textStatus = "The authentication certificate has been succesfully verify through OCSP validation";
else if (status == BEID_CertifStatus.BEID_CERTIF_STATUS_REVOKED)
          textStatus = "The authentication certificate is revoked";
else
          textStatus = "The authentication certificate could not be validated";
```

## The CRL (Certificate Revocation List)

The CRL is automatically downloaded the first time it is required for certificate validation. From then on the MW uses its local cache. However the signing of the CRL is verified before using it. So, if the cache is corrupted, the CRL, will be downloaded again.

The URL of the CRL is embedded in the certificate, so you can get a CRL object directly from the certificate object. Or you can also create a CRL object from an URL. In that case, all information (for example the issuer) may not be available.

With the CRL object, you can easily get the name of the issuer, the issuer certificate object, the CRL data and even write it somewhere on the hard drive. The following snippet shows that last possibility.

Example C++

```
BEID_Crl &crl = certAuth.getCRL();
BEID_ByteArray data;

if (crl.getData(data) == BEID_CRL_STATUS_VALID)
{
          data.writeToFile("MyCrl.crl");
}
```

Example C#

```
BEID_Crl crl = certAuth.getCRL();
BEID_ByteArray data=new BEID_ByteArray();

if (crl.getData(data) == BEID_CrlStatus.BEID_CRL_STATUS_VALID)
{
          data.writeToFile("MyCrl.crl");
}
```

Example Java

```
BEID_Crl crl = certAuth.getCRL();
BEID_ByteArray data = new BEID_ByteArray();

if (crl.getData(data) == BEID_CrlStatus.BEID_CRL_STATUS_VALID)
{
        data.writeToFile("MyCrl.crl");
}
```

## The OCSP (Online Certificate Status Protocol )

Like for the CRL, the OCSP response can be obtained directly from the certificate.

**Remarks:** It's also possible to explicitly ask an OCSP responder for a particular response using a `BEID_OcspResponse` which is not link to a certificate. But in that case you need to provide more information like the URI of the responder, the hash of the issuer name, the serial number of the certificate to check... This feature is not described furthermore here.

From this OCSP response object, you can get the status of the certificate (through OCSP validation). You can also get the entire response data using *getResponse()*. The following snippet illustrates this possibility.

Example C++

```
BEID_OcspResponse &ocsp = certAuth.getOcspResponse();
BEID_ByteArray data;

ocsp.getResponse(data);
data.writeToFile("MyOcsp.ocsp");
```

Example C#

```
BEID_OcspResponse ocsp = certAuth.getOcspResponse();
BEID_ByteArray data = new BEID_ByteArray();

ocsp.getResponse(data);
data.writeToFile("MyOcsp.ocsp");
```

Example Java

```
BEID_OcspResponse ocsp = certAuth.getOcspResponse();
BEID_ByteArray data = new BEID_ByteArray();

ocsp.getResponse(data);
data.writeToFile("MyOcsp.ocsp");
```

# Middleware Settings

The configuration parameters of the middleware are available for reading/writing through the SDK using the *BEID_Config* class. Only the user wide parameters can be changed. (System wide parameters need administrator right to be changed and can't be modified through the SDK).

**Remarks:** You can find more information about all the parameters in the following document: http://eid.belgium.be/fr/binaries/eID3_configparameters_tcm146-22472.pdf

(See also
http://eid.belgium.be/fr/Informations_legales_et_techniques/L_eID_d_un_point_de_vue_technique/ or
http://eid.belgium.be/nl/Achtergrondinfo/De_eID_technisch/ )

You can work with pre-defined parameters or define your own parameters.

There are 2 types of parameters: string and numerical. The following code snippet demonstrates the use of the BEID_Config class:

Example C++

```
BEID_Config param1(BEID_PARAM_GENERAL_LANGUAGE);
const char *language = param1.getString();

BEID_Config param2("MyParam","MySection",0L);
param2.setLong(2);
```

Example C#

```
BEID_Config param1 = new BEID_Config(BEID_Param.BEID_PARAM_GENERAL_LANGUAGE);
string language = param1.getString();

BEID_Config param2 = new BEID_Config("MyParam","MySection",0);
param2.setLong(2);
```

Example Java

```
BEID_Config param1 = new BEID_Config(BEID_Param.BEID_PARAM_GENERAL_LANGUAGE);
String language = param1.getString();

BEID_Config param2 = new BEID_Config("MyParam", "MySection", 0);
param2.setLong(2);
```

# Exceptions

A try/catch mechanism should be put in place in the application where the SDK objects and their methods are used.

All exceptions are derived from the base class *BEID_Exception*. For identification either the exception class type can be used or the exception code the exception carries.

For example, the *BEID_ExNoReader* means that no card readers are connected and the error code assigned to it is *EIDMW_ERR_NO_READER*. The error codes are defined in the file *eidErrors.h*.

Example C++: exception handling:

```
…
try
{
        const char *readerName="test";       // deliberately generate an exception
        reader = &ReaderSet.getReaderByName(readerName);
        …
}
catch(BEID_ExParamRange &ex)
{
…
}
catch(BEID_Exception &ex)
{
…
}
…
```

Example Java: exception handling:

```
…
try
{
        // deliberately generate an exception
        BEID_ReaderContext readerContext = BEID_ReaderSet.instance().getReaderByNum(10);
}
catch (BEID_ExParamRange e)
{
…
}
catch (Exception e)
{
…
}
…
```

Example C#: exception handling:

```
…
try
{
        string readerName="test";     // deliberately generate an exception
        reader = BEID_ReaderSet.instance().getReaderByName(readerName);
        …
}
catch (BEID_ExParamRange ex)
{
…
}
catch(BEID_Exception ex)
{
…
}
…
```

## Logging

For debugging purposes, the SDK also provides a logging function that allows you to add information in the default Middleware log file.

For this function, there is a small difference between the C++ version and other programming languages. Indeed, C++ allows you to pass a format string and an undefined number of parameters. For other programming languages, you must format the message string before passing it to the function.

Example C++

```
BEID_LOG(BEID_LOG_LEVEL_ERROR, "MyModule", "Error code = %ld",2);
```

Example C#

```
beidlib_dotNet.BEID_LOG(BEID_LogLevel.BEID_LOG_LEVEL_ERROR, "MyModule", "Error code = 2");
```

Example Java

```
beidlibJava_Wrapper.BEID_LOG(BEID_LogLevel.BEID_LOG_LEVEL_ERROR, "MyModule", "Error code = 2");
```

# Advanced usage

## Test cards

You can use test card to test and debug your application.

To use test cards, the Middleware must be unlocked at two different levels.

1. In the registry/config, in the section "*certificatevalidation*", it must be a numerical parameter "*cert_allow_testcard*" with the value 1 (in fact non 0).
2. Each test card must be allowed by the user

It's recommended to unlock the first level manually, only on the computer you're using for testing and debugging. If the key in the registry/config doesn't exist (or its value is 0), then the MW will consider the card as an unknown card (like your credit card for example). If the value is 1, the MW will see that it is a Belgian eID test card and after the card is accepted as test card, it will be considered like a normal card.

Once the MW knows it is a test card, the verification of the trusted Belgian root is bypassed.

There are two ways to manage test cards:

- Managed by the application
- Managed by the middleware

### *Manage test cards yourself*

The best practice is not to read test card without explicit consent of the user. This should be done by the following steps:

- check if it is a test card and if it was not allowed yet.
- ask the user if he allows this test card.
- depending on the users response, use the *setAllowTestCard()* function to allow the card or quit your process.

Example C++

```
BEID_EIDCard& Card = ReaderContext.getEIDCard();
if (Card.isTestCard() && !Card.getAllowTestCard())
{
    QString strCaption = tr("Belgian EID Middleware");
    QString strMessage = tr("The Root Certificate is not correct.\nThis may be a test card.\n\nDo you want to accept it?");

    if (QMessageBox::Yes == QMessageBox::warning(this,strCaption,strMessage,QMessageBox::Yes|QMessageBox::No))
    {
        Card.setAllowTestCard(true);
    }
    else
    {
        return;
    }
}
```

**<u>Remark :</u>** This C++ snippet needs Qt

Example C#

```
BEID_EIDCard card = Reader.getEIDCard();
if (card.isTestCard() && !card.getAllowTestCard())
{
    string strCaption     = "Belgian EID Middleware";
    string strMessage     = "The Root Certificate is not correct.\nThis may be a test card.\n\nDo you want to accept it?";

    if (DialogResult.Yes == MessageBox.Show(strMessage, strCaption, MessageBoxButtons.YesNo))
    {
        card.setAllowTestCard(true);
    }
    else
    {
        return;
    }
}
```

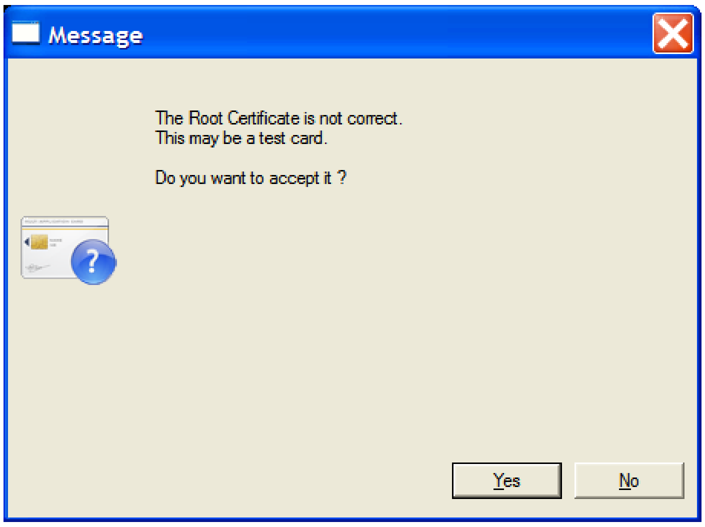
Example Java

```
BEID_EIDCard card = Reader.getEIDCard();
if (card.isTestCard() && !card.getAllowTestCard())
{
    String strCaption = "Belgian EID Middleware";
    String strMessage = "The Root Certificate is not correct.\nThis may be a test card.\n\nDo you want to accept it?";

    if (JOptionPane.YES_OPTION == JOptionPane.showConfirmDialog(null, strMessage, strCaption, JOptionPane.YES_NO_OPTION))
    {
        card.setAllowTestCard(true);
    }
    else
    {
        return;
    }
}
```

## Manage test cards using the middleware

The Middleware can manage the confirmation dialog itself. To do this, the Middleware must be initialized (*initSDK()*) with the parameter set to *true*. Each time a test card is being accessed, the following dialogue will come up:



The Root Certificate is not correct.
This may be a test card.

Do you want to accept it ?

The following snippet shows how to initialize the SDK with the “`ManageTestCard`” parameter.

Example C++

```
BEID_ReaderSet::initSDK(true);
```

Example C#

```
BEID_ReaderSet.initSDK(true);
```

Example Java

```
BEID_ReaderSet.initSDK(true);
```

## Using XML, CSV, TLV format

The SDK allows you to easily save and load documents from the card in 3 different formats: XML, CSV, TLV. This file can be reloaded in a virtual reader.

### *Saving a document*

All the objects that inherit from `BEID_XMLDoc` can be extract in the 3 formats mentioned above. You can use the `getXML()`, `getCSV()` and `getTLV()` method to retrieve the content of such document in a `BEID_ByteArray`. You can also directly save this content in files with the `writeXmlToFile()`, `writeCsvToFile()` and `writeTlvToFile()` methods. Furthermore, you can use the (`getDocument(BEID_DOCTYPE_FULL)` to get all the public data of the card in one *BEID_XMLDoc*, save it somewhere in order to reuse it in a virtual reader in the future.

The following snippet shows how to save the content of a card (eID or SIS) in a xml file.

Example C++

```
BEID_ReaderContext &reader=ReaderSet.getReader();
BEID_Card &card=reader.getCard();
BEID_XMLDoc &doc=card.getDocument(BEID_DOCTYPE_FULL);
doc.writeXmlToFile("MyCard.xml");
```

Example C#

```
BEID_ReaderSet ReaderSet = BEID_ReaderSet.instance();
BEID_ReaderContext reader = ReaderSet.getReader();
BEID_Card card = reader.getCard();
BEID_XMLDoc doc = card.getDocument(BEID_DocumentType.BEID_DOCTYPE_FULL);
doc.writeXmlToFile("MyCard.xml");
```

Example Java

```
BEID_ReaderSet ReaderSet = BEID_ReaderSet.instance();
BEID_ReaderContext reader = ReaderSet.getReader();
BEID_Card card = reader.getCard();
BEID_XMLDoc doc = card.getDocument(BEID_DocumentType.BEID_DOCTYPE_FULL);
doc.writeXmlToFile("MyCard.xml");
```

### *Load a full document in a virtual reader*

With the previously saved full document, we can instantiate a virtual reader and access all the public data like if the card was in a physical reader. Specific operations which need private data, like authentication, signature, PIN verification are not possible with a virtual reader.

The following snippet shows how to create a virtual reader and use this reader context, like a physical reader.

Example C++

```
BEID_ReaderContext reader(BEID_FILETYPE_XML,"MyCard.xml");
```

Example C#

```
BEID_ReaderContext reader = new BEID_ReaderContext(BEID_FileType.BEID_FILETYPE_XML, "c:\\MyCard.xml");
```

Example Java

```
BEID_ReaderContext reader = new BEID_ReaderContext(BEID_FileType.BEID_FILETYPE_XML, "c:\\MyCard.xml");
```

**Remarks:**

The XML and CSV document contain information in 2 different formats: the field and the files. For example, if you open the XML document, you will see a tag <name>. Below, you will see a tag <file_id> and a tag <file_id_sign>. In fact the name included in the tag <name> is for external use but it's not secured. You can trust it if you trust the person how give you the file. Else, the same name is also included in the <file_id> which is signed by the <file_id_sign>. Only these data are secured… (A XML file sample is given below).

The SDK never uses the field data but only the file data. The signatures of the files are verified just like if they would come from a physical card. You can access the content of the field by the appropriate method (*`getName()`* in our use case)

```
<?xml version="1.0" encoding="UTF-8" ?>
- <beid_card>
    <doc_version>1</doc_version>
    <card_type>foreigner</card_type>
  - <biographic>
    - <document>
        <version />
        <type>13</type>
      - <id>
          <name>Katharina Wera</name>
          <surname>Lukas</surname>
          <gender>W</gender>
          <date_of_birth>29.JUN. 1991</date_of_birth>
          <location_of_birth>Köln</location_of_birth>
          <nobility />
          <nationality>Deutschland (Bundesrep.)</nationality>
          <national_nr>91062936836</national_nr>
          <duplicata>00</duplicata>
          <special_organization>1</special_organization>
          <member_of_family />
          <special_status>0</special_status>
        </id>
      + <card>
      </document>
    + <address>
    - <files>
        <file_id encoding="base64">
        AQpCMDAwMDY2MzgxAhBTTElOM2YAKWz/Jw30ARUZAwoyNC4wNS4yMDA3BAoy
        NC4wNS4yMDEyBQpCw7xsbGluZ2VuBgs5MTA2MjkzNjgzNgcFTHVrYXMIDkth
        dGhhcmluYSBXZXJhCQAKGERldXRzY2hsYW5kIChCdW5kZXNyZXAuKQsFS8O2
        bG4MDDI5LkpVTi4gMTk5MQ0BVw4ADwIxMxABMBEUaXTrwqLv2jH5YMpqaLve
        QpAcUuUSAjAwEwExFAA=</file_id>
        <file_id_sign encoding= "base64">
        DXyQVB9rnqWXB/GrTHWYUoSFnc83fyuM+0moXKLc7i/ZUMpzKpD3V0IiYj90
        amXcUqXc7vLykEFxOqLYUVbKmq5WIgghN1h4MGbunBxu88z+ANq9ZBgiY3sp
        ydQW54Jxf10Ym7RECai2jcEWihBOpvMs5Bn0Cw7BQZWqrKY+ZdE=</file_id_sign>
        <file_address encoding="base64">
        ARFCYWhuaG9mc3RyYXNzZSAyMQIENDc2MAMKQsO8bGxpbmdlbgAAAAAAAAAA
        AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA</file_address>
        <file_address_sign encoding="base64">
        UnTZzrU45HHlJrxFoX7aqCHSfHXrTWLMhHNo3juYwx8Is7ToWNwIr4WYdtlD
        HDbHe7MMGekcvGFYGqFdvBX3acMFwJSgUwXpB2KxlrfrA0sbwnEzW66BWSUq
        qAu163dkFyQ5P84P6PqIxtxlFUCcGEj0J+hRVNtqFS+fuSe1BL4=</file_address_sign>
      </files>
    </biographic>
  + <biometric>
  + <scard>
  + <challenge_response>
  + <cryptographic>
</beid_card>
```

## Raw data access

The raw data refers to the files embedded on the card.

The raw data access methods included in the SDK for compatibility reason with previous versions of the Middleware and also to give low level access to the content of the card. For example, this is the only way to access the signature of the id file on the card.

Like for the document, you can access the raw data using a generic method `getRawData(BEID_RawDataType type)` or using specific methods (like `getRawData_IdSig()`) All these methods return a `BEID_ByteArray`.

Our snippet will show how to save the signature of the id file.

Example C++

```
BEID_ByteArray data=card.getRawData_IdSig();
data.writeToFile("id.sig");
```

Example C#

```
BEID_ByteArray data = card.getRawData_IdSig();
data.writeToFile("id.sig");
```

Example Java

```
BEID_ByteArray data = card.getRawData_IdSig();
data.writeToFile("id.sig");
```

**Remarks:**

You can also use the raw data to instantiate a virtual reader using a `BEID_RawData_Eid` or `BEID_RawData_Sis` structure. However, this feature is **deprecated** and it is recommend to work with Xml file instead.

## Using APDU commands

APDU commands are low level commands to access a smart card. Since the eID card is a smartcard, this interface can be used to read data from the eID card.

Since the SDK provides all necessary functionality to read the Belgian eID and SIS cards at a higher level, the use of the APDU commands is unnecessary and strongly discouraged.

Example C++: read the chip serial nr:

```
…
// chip number
unsigned char cmd[] = {0x80, 0xE4, 0x00, 0x00, 0x1C};
BEID_ByteArray cardData(cmd, sizeof(cmd));
BEID_ReaderContext& reader = ReaderSet.getReader();
BEID_EIDCard& card = reader.getEIDCard();
BEID_ByteArray data = card.sendAPDU(cardData);
const unsigned char* serialNr = data.GetBytes();
…
```

Example Java: read the chip serial nr:

```
…
byte[] cmd = { (byte)0x80, (byte)0xE4, (byte)0x00, (byte)0x00, (byte)0x1C };
BEID_ByteArray cardData = new BEID_ByteArray(cmd, 5);
BEID_ReaderContext     reader  = BEID_ReaderSet.instance().getReader();
BEID_EIDCard           card    = reader.getEIDCard();
BEID_ByteArray         data    = card.sendAPDU(cardData);
byte[]                 rawSerialNr = data.GetBytes();
…
```

Example C#: read the chip serial nr:

```
…
byte[] cmd = { 0x80, 0xE4, 0x00, 0x00, 0x1C };
BEID_ByteArray cardData = new BEID_ByteArray(cmd, 5);
BEID_ReaderSet readerSet;
readerSet = BEID_ReaderSet.instance();
BEID_ReaderContext reader;
reader = readerSet.getReader();
BEID_EIDCard card = reader.getEIDCard();
BEID_ByteArray data = card.sendAPDU(cardData);
byte[] rawSerialNr = data.GetBytes();
…
```

# Java Specific issues

## Applet deployment

### *Architecture*

The use of the middleware within an internet applet needs to go through several layers:

1. The html page that embeds the applet.
2. The applet itself (java class that inherits from java.applet.Applet or javax.swing.JApplet).
3. The java MW api "beid35libJava.jar" that gives access to all java classes of the Middleware.
4. The JNI libraries (platform dependent: beid35libJava_Wrapper.dll, libbeidlibJav, Wrapper.jnilib or libbeidlibJava_Wrapper.so) to let the java classes interact with the underlying C++ libraries.
5. The runtime of the middleware (that must be installed separately).

That means that a java web application must take care of the deployment of the java components (Applet, beid35libJava.jar and native java wrapper).
The JNI library to be downloaded depends on the target computer running the applet. To achieve this goal, a special process is needed, being JNLP (Java Network Launching Protocol).

### *JNLP (Java Network Launching Protocol) and Applet-launcher*

To deploy the correct library on the client platform, a generic applet-launcher is used. With the help of a JNLP files, the launcher knows which library it must download for the target computer. The applet-launcher loads the "application applet" known as the 'sub-applet'.

Here is a sample of a JNLP file that maps the target OS to the jar file to be downloaded. The downloaded jar file contains the required native library.

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp codebase="http://127.0.0.1/"
      href="beid.jnlp">
  <information>
    <title>Java Binding to Belgium eID Middleware 3.5</title>
    <vendor>Fedict</vendor>
    <offline-allowed/>
  </information>
  <security>
      <all-permissions/>
  </security>
  <resources os="Windows">
      <nativelib href = "beid35JavaWrapper-win.jar" />
    </resources>
    <resources os="Linux">
      <nativelib href = "beid35JavaWrapper-linux.jar" />
    </resources>
    <resources os="Mac OS X">
      <nativelib href = "beid35JavaWrapper-mac.jar" />
    </resources>
  <component-desc />
</jnlp>
```

The html code to insert the applet-launcher with the JNLP parameter and the sub-applet is shown below.

```
<applet code="org.jdesktop.applet.util.JNLPAppletLauncher"
      codebase = "."
      width  ="140"
      height ="200"
      name   ="BEIDAppletLauncher"
      archive="http://127.0.0.1/applet-launcher.jar,
               http://127.0.0.1/beid35libJava.jar,
               http://127.0.0.1/BEID_Applet.jar">
   <param name="codebase_lookup" value="false">
   <param name="subapplet.classname" value="be.belgium.beid.BEID_Applet">
   <param name="progressbar" value="true">
   <param name="jnlpNumExtensions" value="1">
   <param name="jnlpExtension1" value="http://127.0.0.1/beid.jnlp">

   <param name="debug" value="false"/>
   <param name="Reader" value=""/>
   <param name="OCSP" value="-1"/>
   <param name="CRL" value="-1"/>
 </applet>
```

To be functional, the previous code needs to replace the loopback address (127.0.0.1) with a true URL. If all components are in the same folder, we can make the HTML more generic using some javascript. The following code snippet is another version of the same applet-launcher call as above.

```
<script type="text/javascript" language="javascript">

//----------------
// give the following variables:
// subdir: the subdirectory where the applet jar file resides
// jnlpPath: the path to the jnlp file
// ex: subdir="appletDir";
//     jnlpPath=subdir;
//----------------
var subdir="";
var jnlpPath=subdir;

//----------------
// - get the href of this page
// - strip off the name of this page
//----------------
var myloc    = window.location.href;
var locarray = myloc.split("/");
delete locarray[(locarray.length-1)];
var url = locarray.join("/");

   document.writeln('<applet code="org.jdesktop.applet.util.JNLPAppletLauncher"');
   document.writeln('codebase = "' + url + subdir + '"');
   document.writeln('width  ="140"');
   document.writeln('height ="200"');
   document.writeln('name   = "BEIDAppletLauncher"');
   document.writeln('archive="applet-launcher.jar,beid35libJava.jar,BEID_Applet.jar">');

   document.writeln('<param name="codebase_lookup" value="false">');
   document.writeln('<param name="subapplet.classname" value="be.belgium.beid.BEID_Applet">');
   document.writeln('<param name="progressbar" value="true">');
   document.writeln('<param name="jnlpNumExtensions" value="1">');
   document.writeln('<param name="jnlpExtension1" value= "' + url + jnlpPath + '/beid.jnlp">');

   document.writeln('<param name="debug" value="false"/>');
   document.writeln('<param name="Reader" value=""/>');
   document.writeln('<param name="OCSP" value="-1"/>');
   document.writeln('<param name="CRL" value="-1"/>');
   document.writeln('</applet>');

</script>
```

To access the BEID applet (the sub-applet), the following code must be used:

```
document.BEIDAppletLauncher.getSubApplet()
```

To be sure the applet is launched before using it, we add an onLoad() event on the body section. This event waits until the sub-applet is not null, indicating the applet is loaded.

```
…
<script>

   function bodyLoaded()
   {

      if(document.BEIDAppletLauncher.getSubApplet()==null)
      {
         setTimeout("bodyLoaded()", 100);
      }
      else
      {
         document.BEIDAppletLauncher.getSubApplet().InitLib(null);
      }
   }

</script>
…

<body onLoad = "javascript:bodyLoaded()">
…
</body>
```

In the applet, to load the native library, downloaded using the JNLP process, we need the following function.

```java
private static void loadLibraryInternal(String libraryName)
{
    String sunAppletLauncher = System.getProperty("sun.jnlp.applet.launcher");
    boolean usingJNLPAppletLauncher = Boolean.valueOf(sunAppletLauncher).booleanValue();

    boolean loaded = false;
    if (usingJNLPAppletLauncher)
    {
        try
        {
            Class jnlpAppletLauncherClass= Class.forName("org.jdesktop.applet.util.JNLPAppletLauncher");
            Method jnlpLoadLibraryMethod= jnlpAppletLauncherClass.getDeclaredMethod("loadLibrary",new Class[] {String.class });
            jnlpLoadLibraryMethod.invoke(null, new Object[] { libraryName });
            loaded = true;
        }
        catch (ClassNotFoundException ex)
        {
            System.err.println("loadLibrary(" + libraryName + ")");
            System.err.println(ex);
            System.err.println("Attempting to use System.loadLibrary instead");
        }
        catch (Exception e)
        {
            Throwable t = e;
            if (t instanceof InvocationTargetException)
            {
                t = ((InvocationTargetException)t).getTargetException();
            }
            if (t instanceof Error)
                throw (Error)t;
            if (t instanceof RuntimeException)
            {
                throw (RuntimeException)t;
            }
            // Throw UnsatisfiedLinkError for best compatibility with System.loadLibrary()
            throw (UnsatisfiedLinkError)new UnsatisfiedLinkError().initCause(e);
        }
    }

    if (!loaded)
    {
        System.loadLibrary(libraryName);
    }
}
```

Which we can use as follows.

```java
String osName = System.getProperty("os.name");
String JavaWrapper = "beidlibJava_Wrapper";

if (-1 != osName.indexOf("Windows"))
{
   JavaWrapper = "beid35libJava_Wrapper";
   System.out.println("[Info]  Windows system!!");
   System.out.println("[Info]  Loading Java wrapper: " + JavaWrapper);
   loadLibraryInternal(JavaWrapper);
   System.out.println("[Info]  Java wrapper loaded");
}
else
{
   System.out.println("[Info]  Loading Java wrapper: " + JavaWrapper);
   loadLibraryInternal(JavaWrapper);
   System.out.println("[Info]  Java wrapper loaded");
}
```

**Remarks:** Information on the applet-launcher can be found at: https://applet-launcher.dev.java.net/

### *Jar signature*

To work properly, a java applet must be signed. If the signing certificate is already trusted, the applet runs automatically, otherwise, java requests the user to trust the presented certificate.

In the JNLP process, **the applet-launcher and the jars containing the native libraries must be signed by the same certificate**. The **sub-applet must also be signed,** but this can be a different certificate than the one used for the applet-launcher. In case an other certificate is used, the user will explicitly have to accept both certificates.

All the jars are also available in a government signed version in the folder: beidlib/java/bin.

Using the SDK however, you can build and sign the BEID_Applet.jar yourself.

## Using the picture in html

### *Using the picture data in an HTML page*

The picture on the Belgian eID card is in fact a jpg file. When the picture is read from the eID card, the picture data is held in a *BEID_ByteArray*. This data however cannot be put to an HTML page as raw data.

In order to put the picture data to an html page it must be referenced as “data:image/jpg;base64” in the html page. On the HTML page, either the IMG-tag or the CANVAS-tag can be used as placeholder for the image.

As an example, an applet can be created that is accessing the eID library. A java class is included that does the base64 translation of the data, and some Javascript puts the data to the HTML page.

**Note:** The code in this section is taken from the sample: misc\Applet\Image\java

### *The Applet*

The applet can read the image from the eID card and do the transformation to base64.

Example Java: read picture data and convert in base64:

```
…
public String GetPictureBase64()
{
        try
        {
                byte[] pictRaw = GetPicture();
                String encBytes = Base64.encodeBytes( pictRaw );
                return encBytes;
        }
        catch (Exception e)
        {
                …
        }
        return "";
}
public byte[] GetPicture() throws java.lang.Exception
{
        if (m_card == null)
        {
                return null;
        }
        BEID_Picture picture = m_card.GetPicture();
        if (picture == null)
        {
                return null;
        }
        BEID_ByteArray pictureData = picture.getData();
        byte[] pictureBytes = pictureData.GetBytes();
        try
        {
                Rectangle abounds = getBounds();
                java.awt.Toolkit toolkit = Toolkit.getDefaultToolkit();
                Image tempImage = toolkit.createImage(pictureBytes);
nage.setIcon(new ImageIcon(tempImage.getScaledInstance(abounds.width, abounds.height,
 Image.SCALE_SMOOTH)));
        }
        catch (Exception e)
        {
                …
        }
        return pictureBytes;
}
```

The data from the base64 encoding is returned as a string that can be used by the Javascript to put on the HTML page.

### *Javascript processing*

In Javascript, the Java function *`GetPictureBase64()`* can now be called. The data in base64 must be prepended with 'data:image/jpg:base64' to make sure the browser understands the data format.

It is sufficient to get the IMG-tag element from the HTML page and assign the data to the source. This will 'fill up' the IMG tag and the image appears on the HTML page.

The HTML page could contain both the IMG-tag and the CANVAS-tag.

```
…
<form name=actionform>
                <applet
                  codebase = "."
                  archive  = "BEID_ImgApplet.jar,beid35libJava.jar"
                  code     = "be.belgium.beid.BEID_ImgApplet.class"
                  name     = "BEIDApplet"
                  width    = "140"
                  height   = "200"
                  hspace   = "0"
                  vspace   = "0"
                  align    = "middle"
                >
                </applet>
               <p>
               <input type="button" name="IDButton"
       onclick="javascript:readCard()"      value="Read Card " title="Read Card"/>
               <select id="ID_CardReadersCombo">
               </select>
               <p>
               <br>
               <canvas id="theCanvas" width="140" height="200" alt="Not supported by this
browser"></canvas>
               <p>
               <br>
               <img id="theImage" width="140" height="200" alt="Not supported by this
browser"/>
</form>
…
```

Example Java: Javascript to put the image to an IMG-tag on an HTML page:

```
…
var pictEncoded = document.BEIDApplet.GetPictureBase64();
eIDImage = new Image();
eIDImage.src = "data:image/jpg;base64,"+ pictEncoded;
var img = document.getElementById('theImage');
img.src = eIDImage.src;
…
```

For the CANVAS-tag, the Javascript code is different:

Example Java: Javascript to put the image to a CANVAS-tag on an HTML page:

```
function loadCanvas()
{
        if(eIDImage.complete)
        {
                var canvas = document.getElementById('theCanvas');
                if (canvas.getContext)
                {
                        var ctx = canvas.getContext('2d');
                        ctx.drawImage(eIDImage, 0, 0);
                }
                else
                {
                        alert( "Canvas not supported" );
                }
        }
}
```

Remark that first of all a test is done to see that the image is completed for the eIDImage object. This is necessary in case the IMG tag is processed first. The browser loads the image and indicates it has finished by putting the '*complete*' property to true. Once this is accomplished, the CANVAS-tag can be processed.

### *Browser compatibility*

Not all browsers support the CANVAS-tag. The table below gives an overview:

Windows:

| **Browser** | **Applet image** | **IMG tag base64 data** | **CANVAS tag base64 data** |
|---|---|---|---|
| IE 7 | Yes | No | No |
| IE 8(beta) | Yes | Yes | No |
| Firefox | Yes | Yes | Yes |
| Safari | Yes | Yes | Yes |
| Google Chrome | Yes | Yes | Yes |
| Opera | Yes | Yes | Yes |

Remarks:

- IE8 needs a special javascript (excanvas.js) to enable the CANVAS tag
- IE8 can handle an IMG tag with data in base64 encoding. The canvas tag does not work in IE8 with base64 encoded data.
- JRE to be used depends on the browser (>= 1.5)

Mac OS X:

- Firefox, Safari and Opera allow the base64 encoded data and CANVAS-tag

Linux:

- Firefox, Opera allow the base64 encoded data and CANVAS-tag

# Cryptographic modules

Cryptographic modules allow signing by different applications (like MS-Word, OpenOffice,…) or to authenticate to secure websites with a web browser.

Each time it is required, the middleware will automatically ask to enter your pin code.

The pin code is requested:

- each time you need to use the private signature key (for example to sign a document).
- once per session for the private authentication key usage. To be more precise, the single sign on mechanism (internal to the card), retains the pin until the card is powered off.

## PKCS#11

PKCS#11 is a standard multiplatform cryptographic API. It allows you to get information about a cryptographic object on a smart card, use the smart card to sign data, …

**Remarks:** This document does not describe the whole PKCS#11 API, but just shows how to use the SDK for signing data (samples in C++, Java and C#).

In the following code snippet, we will sign data, using the private signature key from the card and save the signature into a file. (The signed data is the executable binary itself).

These steps explain the signing process

- The PKCS#11 library is loaded dynamically.
- From *C_GetFunctionList()*, we can access all the other functions.
- *C_Initialize()* prepares the usage of the API.
- *C_GetSlotList()* retrieves the slot (= reader) with a token (= card). (For simplicity we just take the first one).
- *C_OpenSession()* initiate the communication with this slot/token (= reader/card).
- Then we get a handle to the signature private key. We use the *C_FindObjects()* function to retrieve the object of class `CKO_PRIVATE_KEY` with the id 3. This is the id of the signature key on Belgium eID card. The private authentication key has the id 2.
- *C_SignInit()* prepares the signing with the mechanism `CKM_SHA1_RSA_PKCS`, meaning we do the hash and signature in a single step.
- Then, we load the data and pass them to the *C_Sign()* function which returns the signature.
- Finally, after writing out the signature, we can close de session and finalize the PKCS#11 API usage.

## Example C++

```cpp
#include <iostream>
#include <windows.h>
#include <sys/types.h>
#include <sys/stat.h>
#include "cryptoki.h"   //Download from http://www.rsa.com/rsalabs/node.asp?id=2133

typedef CK_RV (*P11_GetFunctionList)(CK_FUNCTION_LIST_PTR_PTR ppFunctionList);
CK_RV LastRV=CKR_OK;

const char* signPkcs11( ) {
    HMODULE hLibrary=LoadLibrary(L"beidpkcs11.dll");
    if(!hLibrary)                                                      return "Library not found";

    P11_GetFunctionList getP11Function = (P11_GetFunctionList) GetProcAddress(hLibrary, "C_GetFunctionList");
    CK_FUNCTION_LIST_PTR p11;
    getP11Function(&p11);

    //Initialize PKCS#11
    LastRV=p11->C_Initialize(NULL);                                    if(LastRV != CKR_OK) return "C_Initialize failed.";

    //Get the list of reader with card (we are only interested with the first one)
    CK_SLOT_ID p11_slot=0;
    CK_ULONG p11_num_slots=1;
    LastRV=p11->C_GetSlotList(TRUE, &p11_slot, &p11_num_slots);        if(LastRV != CKR_OK) return "C_GetSlotList failed.";

    //Open the P11 session
    CK_SESSION_HANDLE p11_session = CK_INVALID_HANDLE;
    LastRV=p11->C_OpenSession(p11_slot,CKF_SERIAL_SESSION | CKF_RW_SESSION,NULL, NULL, &p11_session);
                                                                       if(LastRV != CKR_OK) return "C_OpenSession failed.";
    //Find the signature private key
    CK_ATTRIBUTE attrs[2];
    CK_OBJECT_CLASS cls = CKO_PRIVATE_KEY;
    CK_ULONG id=3;
    CK_OBJECT_HANDLE signaturekey=CK_INVALID_HANDLE;
    CK_ULONG count;

    attrs[0].type = CKA_CLASS;
    attrs[0].pValue = &cls;
    attrs[0].ulValueLen = sizeof(cls);
    attrs[1].type = CKA_ID;
    attrs[1].pValue = &id;
    attrs[1].ulValueLen = sizeof(id);

    LastRV=p11->C_FindObjectsInit(p11_session,attrs,2);                if(LastRV != CKR_OK) return "C_FindObjectsInit failed.";
    LastRV=p11->C_FindObjects(p11_session, &signaturekey, 1, &count);  if(LastRV != CKR_OK) return "C_FindObjects failed.";
    if(count==0)                                                       return "Signature key not found.";
    LastRV=p11->C_FindObjectsFinal(p11_session);                       if(LastRV != CKR_OK) return "C_FindObjectsFinal failed.";

    //Initialize the signature
    CK_MECHANISM    mech;
    mech.mechanism=CKM_SHA1_RSA_PKCS;
    mech.pParameter=NULL;
    mech.ulParameterLen=0;
    LastRV=p11->C_SignInit(p11_session, &mech, signaturekey);          if(LastRV != CKR_OK) return "C_SignInit failed.";

    //Open the data to sign
    unsigned char signature[1024];
    CK_ULONG sig_len = sizeof(signature);
    unsigned char *data=NULL;
    CK_ULONG data_len=0;
    FILE *f=NULL;

    f = fopen("sign_p11.exe", "rb");
    if (f == NULL)                                                     return "Could not find file sign_p11.exe";
    struct _stat file_info = {0};
    if(0 == _fstat(_fileno(f), &file_info)) {
        data = (unsigned char *)malloc(file_info.st_size);
        data_len = file_info.st_size;
        if(data_len != fread(data, sizeof(unsigned char), data_len, f))
        { free(data);                                                  return "Problem reading file sign_p11.exe"; }
    }
    fclose(f);
    f=NULL;

    //Sign the data
    LastRV=p11->C_Sign(p11_session, data, data_len, signature, &sig_len);  if(LastRV != CKR_OK) return "C_Sign failed.";
    free(data);

    //Write the signature into file
    f = fopen("sign_p11_cpp.sig", "wb");
    if (f == NULL)                                                     return "Could not create sign_p11.sig";
    if(sig_len != fwrite(signature, sizeof(unsigned char), sig_len, f))return "Problem writing file sign_p11.sig";
    fclose(f);
    f=NULL;

    //Close the session and Finalize
    LastRV=p11->C_CloseSession(p11_session);                           if(LastRV != CKR_OK) return "C_CloseSession failed.";
    LastRV=p11->C_Finalize(NULL);                                      if(LastRV != CKR_OK) return "C_Finalize failed.";
    FreeLibrary(hLibrary);

    return "SUCCEED";
}

int main( int argc, char* argv[] ) {
…
    const char *signMsg=signPkcs11();                                  std::cout << "Signature: " << signMsg << std::endl;
    if(LastRV)                                                         std::cout << "LastRv=" << std::hex << LastRV << std::endl;
…
}
```

Example C#

To use PKCS#11 with C#, we need to access the C functions from C#. For simplicity reasons, the following code is not a full wrapper around the C API, it just wraps the functions and declarations we need to achieve our goal.

**Remarks:** As this sample shows, it's possible to use PKCS#11 with C# code. However, as C# is mostly related to windows, C# user may prefer to use CSP above PKCS#11(see paragraph CSP below)

To make usage and understanding more easy, we first redefine the PKCS#11 types.

```
using CK_ULONG = System.UInt32;
using CK_VOID_PTR = System.IntPtr;
using CK_BBOOL = System.Boolean;
using CK_SLOT_ID = System.UInt32;           //CK_ULONG;
using CK_OBJECT_CLASS = System.UInt32;      //CK_ULONG;
using CK_SESSION_HANDLE = System.UInt32;    //CK_ULONG;
using CK_FLAGS = System.UInt32;             //CK_ULONG;
using CK_NOTIFY = System.IntPtr;            //Callback function, not used in this sample
using CK_OBJECT_HANDLE = System.UInt32;     //CK_ULONG;
using CK_ATTRIBUTE_TYPE = System.UInt32;    //CK_ULONG;
using CK_MECHANISM_TYPE = System.UInt32;    //CK_ULONG;
using CK_BYTE = System.Byte;
```

As we need to pass an array of C structures to the *C_FindObjectsInit()* function, we included a C# Memory class to recreate this parameter in unmanaged memory. (We use a sample class from Microsoft in which we add the following *Append* methods.)

```
public unsafe class Memory //Code from http://msdn.microsoft.com/en-us/library/aa664786(VS.71).aspx
{

…

    // Copies count bytes from src to dst.
    // and increment the dst pointer
    public static void Append(void* src, void* dst, int count)
    {
        Copy(src, dst, count);
        dst = (void*)((int)dst + count);
    }
    public static void Append(uint srcIn, ref void* dst)
    {
        int count = sizeof(uint);
        void* src = &srcIn;
        Copy(src, dst, count);
        dst = (void*)((int)dst + count);
    }
    public static void Append(uint* srcIn, ref void* dst)
    {
        int count = sizeof(uint*);
        void* src = &srcIn;
        Copy(src, dst, count);
        dst = (void*)((int)dst + count);
    }

…

}
```

Then we can define the constants and the prototypes of the external functions we will use.

```
const string p11Library = "beidpkcs11.dll";

const uint CKR_OK = 0;

const uint CK_INVALID_HANDLE=0;

const uint CKF_RW_SESSION = 2;
const uint CKF_SERIAL_SESSION = 4;

const uint CKO_PRIVATE_KEY = 3;

const uint CKA_CLASS = 0;
const uint CKA_ID = 0x00000102;

const uint CKM_SHA1_RSA_PKCS = 0x00000006;

[StructLayout(LayoutKind.Sequential)]
unsafe struct CK_ATTRIBUTE
{
    public CK_ATTRIBUTE_TYPE type;
    public void* pValue;
    public CK_ULONG ulValueLen;
}

[StructLayout(LayoutKind.Sequential)]
struct CK_MECHANISM {
    public CK_MECHANISM_TYPE mechanism;
    public CK_VOID_PTR pParameter;          //This parameter is not used here, so unsafe struct is not needed
    public CK_ULONG ulParameterLen;
}

[DllImport(p11Library, EntryPoint = "C_Initialize", CharSet = CharSet.Ansi, SetLastError = true)]
static extern uint C_Initialize(CK_VOID_PTR param);

[DllImport(p11Library, EntryPoint = "C_Finalize", CharSet = CharSet.Ansi, SetLastError = true)]
static extern uint C_Finalize(CK_VOID_PTR reserved);

[DllImport(p11Library, EntryPoint = "C_GetSlotList", CharSet = CharSet.Ansi, SetLastError = true)]
static extern uint C_GetSlotList(CK_BBOOL tokenPresent, ref CK_SLOT_ID pSlotList, ref CK_ULONG pulCount);

[DllImport(p11Library, EntryPoint = "C_OpenSession", CharSet = CharSet.Ansi, SetLastError = true)]
static extern uint C_OpenSession(CK_SLOT_ID slotID, CK_FLAGS flags, CK_VOID_PTR pApplication, CK_NOTIFY Notify, out CK_SESSION_HANDLE phSession);

[DllImport(p11Library, EntryPoint = "C_CloseSession", CharSet = CharSet.Ansi, SetLastError = true)]
static extern uint C_CloseSession(CK_SESSION_HANDLE hSession);

[DllImport(p11Library, EntryPoint = "C_FindObjectsInit", CharSet = CharSet.Ansi, SetLastError = true)]
unsafe static extern uint C_FindObjectsInit(CK_SESSION_HANDLE hSession, CK_ATTRIBUTE* pTemplate, CK_ULONG ulCount);

[DllImport(p11Library, EntryPoint = "C_FindObjects", CharSet = CharSet.Ansi, SetLastError = true)]
static extern uint C_FindObjects(CK_SESSION_HANDLE hSession, out CK_OBJECT_HANDLE phObject, CK_ULONG ulMaxObjectCount, ref CK_ULONG
pulObjectCount);

[DllImport(p11Library, EntryPoint = "C_FindObjectsFinal", CharSet = CharSet.Ansi, SetLastError = true)]
static extern uint C_FindObjectsFinal(CK_SESSION_HANDLE hSession);

[DllImport(p11Library, EntryPoint = "C_SignInit", CharSet = CharSet.Ansi, SetLastError = true)]
static extern uint C_SignInit(CK_SESSION_HANDLE hSession,ref CK_MECHANISM  pMechanism, CK_OBJECT_HANDLE hKey);

[DllImport(p11Library, EntryPoint = "C_Sign", CharSet = CharSet.Ansi, SetLastError = true)]
static extern uint C_Sign(CK_SESSION_HANDLE hSession, CK_BYTE[] pData, CK_ULONG ulDataLen, IntPtr pSignature, ref CK_ULONG pulSignatureLen);
```

Finally, there is the code for the signing. The flow is the same as for C++. The only differences are in populating the parameters to let them be understood by the C functions.

```
uint LastRV = CKR_OK;

//Initialize PKCS#11
LastRV = C_Initialize(CK_VOID_PTR.Zero);
if (LastRV != CKR_OK) { MessageBox.Show("C_Initialize failed.\n\n(Error=" + LastRV.ToString("x") + ")"); return; }

//Get the list of reader with card (we are only interested with the first one)
CK_SLOT_ID p11_slot=0;
CK_ULONG p11_num_slots=1;
LastRV = C_GetSlotList(true, ref p11_slot, ref p11_num_slots);
if (LastRV != CKR_OK) { MessageBox.Show("C_GetSlotList failed.\n\n(Error=" + LastRV.ToString("x") + ")"); return; }

//Open the P11 session
CK_SESSION_HANDLE p11_session = CK_INVALID_HANDLE;
LastRV = C_OpenSession(p11_slot, CKF_SERIAL_SESSION | CKF_RW_SESSION, CK_VOID_PTR.Zero, CK_VOID_PTR.Zero, out p11_session);
if (LastRV != CKR_OK) { MessageBox.Show("C_OpenSession failed.\n\n(Error=" + LastRV.ToString("x") + ")"); return; }

//Find the signature private key
unsafe
{
    try
    {
        CK_OBJECT_CLASS cls = CKO_PRIVATE_KEY;
        CK_ULONG id = 3;

        int struct_size = 2 * sizeof(CK_ATTRIBUTE);
        void* pAttrs = Memory.Alloc(struct_size);
        void* pAppend = pAttrs;

        Memory.Append(CKA_CLASS, ref pAppend);                 //attrs[0].type
        Memory.Append(&cls, ref pAppend);                      //attrs[0].pValue
        Memory.Append(sizeof(CK_OBJECT_CLASS), ref pAppend);  //attrs[0].ulValueLen

        Memory.Append(CKA_ID, ref pAppend);                    //attrs[1].type
        Memory.Append(&id, ref pAppend);                       //attrs[1].pValue
        Memory.Append(sizeof(CK_ULONG), ref pAppend);         //attrs[1].ulValueLen

        LastRV = C_FindObjectsInit(p11_session, (CK_ATTRIBUTE*)pAttrs, 2);
        Memory.Free(pAttrs);
    }
    catch(OutOfMemoryException ex)
    { MessageBox.Show("Out of memory."); return; }
    catch (InvalidOperationException ex)
    { MessageBox.Show("Invalid memory operation."); return; }
    catch
    { MessageBox.Show("Invalid memory operation."); return; }
}
if (LastRV != CKR_OK) { MessageBox.Show("C_FindObjectsInit failed.\n\n(Error=" + LastRV.ToString("x") + ")"); return; }

CK_OBJECT_HANDLE signaturekey=CK_INVALID_HANDLE;
CK_ULONG count=0;
LastRV=C_FindObjects(p11_session, out signaturekey, 1, ref count);
if (LastRV != CKR_OK) { MessageBox.Show("C_FindObjects failed.\n\n(Error=" + LastRV.ToString("x") + ")"); return; }

if(count==0)        { MessageBox.Show("Signature key not found."); return; }

LastRV=C_FindObjectsFinal(p11_session);
if (LastRV != CKR_OK) { MessageBox.Show("C_FindObjectsFinal failed.\n\n(Error=" + LastRV.ToString("x") + ")"); return; }

//Open the data to sign
byte[] data;
uint data_len = 0;

string fileToSign = "sign_P11.exe";

if (!System.IO.File.Exists(fileToSign))
{ MessageBox.Show("Could not find file " + fileToSign); return; }

//Initialize the signature
CK_MECHANISM mech;
mech.mechanism = CKM_SHA1_RSA_PKCS;
mech.pParameter = IntPtr.Zero;
mech.ulParameterLen = 0;
LastRV = C_SignInit(p11_session, ref mech, signaturekey);
if (LastRV != CKR_OK) { MessageBox.Show("C_SignInit failed.\n\n(Error=" + LastRV.ToString("x") + ")"); return; }

data = System.IO.File.ReadAllBytes(fileToSign);
data_len = (uint)data.GetLength(0);

//Sign the data
uint sig_len = 1024;
IntPtr ptSignature = Marshal.AllocHGlobal((int)sig_len);

LastRV = C_Sign(p11_session, data, data_len, ptSignature, ref sig_len);
if (LastRV != CKR_OK) { MessageBox.Show("C_Sign failed.\n\n(Error=" + LastRV.ToString("x") + ")"); return; }

byte[] signature = new byte[sig_len];
Marshal.Copy(ptSignature, signature, 0, (int)sig_len);

//Write the signature into a file
System.IO.File.WriteAllBytes("sign_p11_cs.sig", signature);

Marshal.FreeHGlobal(ptSignature);

//Close the session
LastRV = C_CloseSession(p11_session);
if (LastRV != CKR_OK) { MessageBox.Show("C_CloseSession failed.\n\n(Error=" + LastRV.ToString("x") + ")"); return; }

//Finalize PKCS#11
LastRV = C_Finalize(CK_VOID_PTR.Zero);
if (LastRV != CKR_OK) { MessageBox.Show("C_Finalize failed.\n\n(Error=" + LastRV.ToString("x") + ")"); return; }

MessageBox.Show("Signing successful");
```

## Example Java

Java6 directly included a PKCS#11 wrapper (in the sun.security.pkcs11.wrapper package) which makes it very easy to use.

```java
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.File;
import java.io.FileInputStream;
import java.io.DataInputStream;
import java.io.FileOutputStream;
import java.io.DataOutputStream;

import sun.security.pkcs11.wrapper.CK_ATTRIBUTE;
import sun.security.pkcs11.wrapper.CK_MECHANISM;
import sun.security.pkcs11.wrapper.PKCS11;
import sun.security.pkcs11.wrapper.PKCS11Constants;
import sun.security.pkcs11.wrapper.PKCS11Exception;

public class sign_p11
{
    public static void main(String[] args) throws IOException, PKCS11Exception
    {
        PKCS11 pkcs11;

        String osName = System.getProperty("os.name");

        try
        {

            if (-1 != osName.indexOf("Windows"))
                pkcs11 = PKCS11.getInstance("beidpkcs11.dll", "C_GetFunctionList", null, false);
            else
                pkcs11 = PKCS11.getInstance("libbeidpkcs11.so","C_GetFunctionList", null, false);

            //Open the P11 session
            long p11_session = pkcs11.C_OpenSession(0, PKCS11Constants.CKF_SERIAL_SESSION, null, null);

            try
            {
                //Find the signature private key
                CK_ATTRIBUTE[] attributes = new CK_ATTRIBUTE[2];
                attributes[0] = new CK_ATTRIBUTE();
                attributes[0].type = PKCS11Constants.CKA_CLASS;
                attributes[0].pValue = new Long(PKCS11Constants.CKO_PRIVATE_KEY);
                attributes[1] = new CK_ATTRIBUTE();
                attributes[1].type = PKCS11Constants.CKA_ID;
                attributes[1].pValue = 3;

                pkcs11.C_FindObjectsInit(p11_session, attributes);

                long[] keyHandles = pkcs11.C_FindObjects(p11_session, 1);
                long signatureKey = keyHandles[0];

                pkcs11.C_FindObjectsFinal(p11_session);

                //Open the data to sign
                File file = null;
                file = new File ("sign_p11.class");
                FileInputStream file_input = new FileInputStream (file);
                DataInputStream data_in    = new DataInputStream (file_input);

                byte[] data = new byte[(int)file.length()];
                data_in.read(data);
                data_in.close();

                //Initialize the signature
                CK_MECHANISM mechanism = new CK_MECHANISM();
                mechanism.mechanism = PKCS11Constants.CKM_SHA1_RSA_PKCS;
                mechanism.pParameter = null;
                pkcs11.C_SignInit(p11_session, mechanism, signatureKey);

                //Sign the data
                byte[] signature = pkcs11.C_Sign(p11_session, data);

                //Write the signature into a file
                file = new File ("sign_p11_java.sig");
                FileOutputStream file_output = new FileOutputStream (file);
                DataOutputStream data_out = new DataOutputStream (file_output);
                data_out.write(signature);
                file_output.close();

                System.out.println("Signing successful");
            }
            catch (Exception e)
            {
                System.out.println("[Catch] Exception: " + e.getMessage());
            }
            finally
            {
                //Close the session
                pkcs11.C_CloseSession(p11_session);
            }
        }
        catch (Exception e)
        {
            System.out.println("[Catch] Exception: " + e.getMessage());
        }
    }
}
```

# CSP

CSP (Cryptographic Service Provider) is a Microsoft technology to do cryptographic operations.

Unlike PKCS#11, CSP is based on the Windows certificate store. This makes the registration of the certificate a prerequisite to the usage of the card with CSP.

The Belgian eID Middleware registers the certificates in the Windows store, using the following conventions:

- The provider name is : “Belgium Identity Card CSP”
- The container name are : <CertificateType>(<CardSerialNumber>).

<CertificateType> is “Signature” or “Authentication” regarding which certificate we want to use.

<CardSerialNumber> is the 16 bytes serial number of the card in a 32 hexadecimal characters format.

For example, “Signature(534C494E336600296CFF270DF4011519)” is a valid container name.

Combining the MW to register the certificate, the beidlib SDK to get information on the inserted card and the CSP API for signing, we can easily create signatures. (See the following snippet).

The signing process follows these steps:

- First, we use the *getReader()* method without any argument to retrieve the first reader containing a card.
- Then we ask for the *BEID_CardVeriosnInfo()* document which will give us access to the serial number of the card.
- The CSP usage always begins with a `CryptAcquireContextA()` call.
- The signature is made by creating a hash with `CryptCreateHash()`, populate it with `CryptHashData()`, and signing this hash with `CryptSignHash()`.
- Finally we terminate with some clean up: `CryptDestroyHash()` and, `CryptReleaseContext()`.

**Remarks:**

- The registration of the certificate in the windows store can be made by the Belgian eID middleware and is out of the scope of this document.
- The code snippet does not show any “card management”. In a normal application, we should check if the card is present and if it is a valid Belgian eID card.

Example C++

```
#include <iostream>
#include <windows.h>
#include <sys/types.h>
#include <sys/stat.h>
#include "eidlib.h"

using namespace eIDMW;

int LastError=0;

const char* signCSP(const char *card_serial_number )
{
    HCRYPTPROV hprov = NULL;

    //Get a context
    if (! CryptAcquireContextA(&hprov, card_serial_number, "Belgium Identity Card CSP", PROV_RSA_FULL, CRYPT_VERIFYCONTEXT))
                                                                   { LastError = GetLastError(); return "CryptAcquireContext failed."; }

    //Open the data to sign
    unsigned char signature[1024];
    unsigned long sig_len = sizeof(signature);
    unsigned char *data=NULL;
    unsigned long data_len=0;
    FILE *f=NULL;

    f = fopen("sign_csp.exe", "rb");

    if (f == NULL)                                                         return "Could not find file sign_csp.exe";
    struct _stat file_info = {0};
    if(0 == _fstat(_fileno(f), &file_info))
    {
        data = (unsigned char *)malloc(file_info.st_size);
        data_len = file_info.st_size;
        if(data_len != fread(data, sizeof(unsigned char), data_len, f))
        {
            free(data);                                                    return "Problem reading file sign_csp.exe";
        }
    }
    fclose(f);
    f=NULL;

    HCRYPTHASH hhash;

    if (!CryptCreateHash(hprov, CALG_SHA1, 0, 0, &hhash))              { LastError = GetLastError(); return "CryptCreateHash failed."; }

    if (!CryptHashData(hhash, data, data_len, 0))                          { LastError = GetLastError(); return "CryptHashData failed"; }

    if (!CryptSignHash(hhash, AT_SIGNATURE, NULL, 0, signature, &sig_len)) { LastError = GetLastError();return "CryptSignHash failed."; }

    //Write the signature into file
    f = fopen("sign_csp_cpp.sig", "wb");
    if (f == NULL)                                                         return "Could not create sign_csp.sig";

    if(sig_len != fwrite(signature, sizeof(unsigned char), sig_len, f))    return "Problem writing file sign_csp.sig";

    fclose(f);
    f=NULL;

    if (NULL != data)   free(data);
    if (NULL != hhash)  CryptDestroyHash(hhash);
    if (NULL != hprov)  CryptReleaseContext(hprov, 0);

    return "SUCCEED";
}

int main( int argc, char* argv[] )
{
    std::cout << "Please insert a card to sign.” << std::endl;
    std::cout << "Be sure that certificate are registered in the windows store. << std::endl;
    std::cout << "(Press enter when ready)" << std::endl;

    char c=getchar();

    BEID_InitSDK();
    BEID_ReaderContext &reader = ReaderSet.getReader();
    BEID_EIDCard &card = reader.getEIDCard();
    BEID_CardVersionInfo &doc = card.getVersionInfo();

    char card_serial_number[50];
    sprintf(card_serial_number,"Signature(%s)",doc.getSerialNumber());

    std::cout << "Container = " << card_serial_number << std::endl;

    const char *signMsg=signCSP(card_serial_number);
    std::cout << "Signature: " << signMsg << std::endl;

    if(LastError) std::cout << "LastError=" << std::hex << LastError << std::endl;

    BEID_ReleaseSDK();

}
```

Example C#

Like for PKCS#11, we need to wrap the CSP function we use. The following code is not a complete wrapper and for simplicity, it just gives access to the needed functions.

First we begin by redefining some specific types.

```
using DWORD = System.UInt32;

using HCRYPTPROV = System.UInt32;
using HCRYPTKEY = System.UInt32;
using HCRYPTHASH = System.UInt32;
using ALG_ID = System.UInt32;
```

Then we redefine the constants and decorate the C functions.

```
const uint PROV_RSA_FULL = 1;
const uint AT_SIGNATURE = 2;

const uint CRYPT_VERIFYCONTEXT    = 0xF0000000;

const uint CALG_SHA1  = 0x00008004;

[DllImport(@"advapi32.dll", EntryPoint = "CryptAcquireContextA",CharSet = CharSet.Ansi, SetLastError = true)]
internal static extern bool CryptAcquireContext(out HCRYPTPROV phProv, string pszContainer, string pszProvider,
DWORD dwProvType, DWORD dwFlags);

[DllImport(@"advapi32.dll", EntryPoint = "CryptReleaseContext",CharSet = CharSet.Ansi, SetLastError = true)]
internal static extern bool CryptReleaseContext(HCRYPTPROV hProv, DWORD dwFlags);

[DllImport(@"advapi32.dll", EntryPoint = "CryptCreateHash",CharSet = CharSet.Ansi, SetLastError = true)]
internal static extern bool CryptCreateHash(HCRYPTPROV hProv, ALG_ID Algid, HCRYPTKEY hKey, DWORD dwFlags, out
HCRYPTHASH phHash);

[DllImport(@"advapi32.dll", EntryPoint = "CryptDestroyHash",CharSet = CharSet.Ansi, SetLastError = true)]
internal static extern bool CryptDestroyHash(HCRYPTHASH hHash);

[DllImport(@"advapi32.dll", EntryPoint = "CryptHashData",CharSet = CharSet.Ansi, SetLastError = true)]
internal static extern bool CryptHashData(HCRYPTHASH hHash, byte[] pbData, DWORD dwDataLen, DWORD dwFlags);

[DllImport(@"advapi32.dll", EntryPoint = "CryptSignHashA",CharSet = CharSet.Ansi, SetLastError = true)]
internal static extern bool CryptSignHash(HCRYPTHASH hHash, DWORD dwKeySpec, string sDescription, DWORD dwFlags,
IntPtr pbSignature, ref DWORD pdwSigLen);
```

Finally, we get the serial number of the card and create the signature. The process is the same as in C++.

```
//Get the name of the container
BEID_ReaderContext reader = BEID_ReaderSet.instance().getReader();
BEID_EIDCard card = reader.getEIDCard();
BEID_CardVersionInfo doc = card.getVersionInfo();

string container = "Signature(" + doc.getSerialNumber() + ")";
string provider = "Belgium Identity Card CSP";

HCRYPTPROV hprov;

//Get a context
if (!CryptAcquireContext(out hprov, container, provider, PROV_RSA_FULL, CRYPT_VERIFYCONTEXT))
{
    MessageBox.Show("CryptAcquireContext failed.\n\n(Error=" + Marshal.GetLastWin32Error().ToString("x") + ")");
    return;
}

//Open the data to sign
byte[] data;
uint data_len=0;

string fileToSign = "sign_csp.exe";

if (!System.IO.File.Exists(fileToSign))
{
    MessageBox.Show("Could not find file " + fileToSign );
   return;
}
data = System.IO.File.ReadAllBytes(fileToSign);
data_len = (uint)data.GetLength(0);

HCRYPTHASH hhash;

//Get an handle to a new hash
if (!CryptCreateHash(hprov, CALG_SHA1, 0, 0, out hhash))
{
    MessageBox.Show("CryptCreateHash failed.\n\n(Error=" + Marshal.GetLastWin32Error().ToString("x") + ")");
    return;
}

//Create the hash
if (!CryptHashData(hhash, data, data_len, 0))
{
    MessageBox.Show("CryptHashData failed.\n\n(Error=" + Marshal.GetLastWin32Error().ToString("x") + ")");
    return;
}

uint sig_len = 1024;
IntPtr ptSignature = Marshal.AllocHGlobal((int)sig_len);

//Sign the hash
if (!CryptSignHash(hhash, AT_SIGNATURE, "", 0, ptSignature, ref sig_len))
{
    MessageBox.Show("CryptSignHash failed.\n\n(Error=" + Marshal.GetLastWin32Error().ToString("x") + ")");
    return;
}
byte[] signature = new byte[sig_len];
Marshal.Copy(ptSignature, signature, 0, (int)sig_len);

//Write the signature into a file
System.IO.File.WriteAllBytes("sign_csp_cs.sig", signature);

Marshal.FreeHGlobal(ptSignature);

//Release hash
if (!CryptDestroyHash(hhash))
{
    MessageBox.Show("CryptDestroyHash failed.\n\n(Error=" + Marshal.GetLastWin32Error().ToString("x") + ")");
     return;
}

//Release context
if (!CryptReleaseContext(hprov, 0))
{
    MessageBox.Show("CryptReleaseContext failed.\n\n(Error=" + Marshal.GetLastWin32Error().ToString("x") + ")");
    return;
}

MessageBox.Show("Signing successful");
```